

```

# =====
# 1) IMPORT LIBRARIES
# =====

import pandas as pd
import numpy as np

from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, classification_report

import seaborn as sns
import matplotlib.pyplot as plt

# =====
# 2) FEATURE ENGINEERING
# =====

# محمل مسبقاً df_hosp يفترض أن
# إجمالي الزيارات
df_hosp["total_visits"] = (
    df_hosp["number_outpatient"] +
    df_hosp["number_emergency"] +
    df_hosp["number_inpatient"]
)

# الإجراءات الطبية الكبرى
df_hosp["major_procedures"] = (
    df_hosp["num_lab_procedures"] +
    df_hosp["num_procedures"]
)

# A1C mapping
map_a1c = {"None": 0, ">7": 1, ">8": 2}
df_hosp["A1C_num"] = df_hosp["A1Cresult"].map(map_a1c)

# Glucose mapping
map_glu = {"None": 0, "Normal": 1, ">200": 2, ">300": 3}
df_hosp["glu_num"] = df_hosp["max_glu_serum"].map(map_glu)

# العمر
df_hosp["age_clean"] = df_hosp["age"].str.extract(r"(\d+)").astype(float)

# عدد الأدوية
drug_cols = [
    'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride'

```

```

,
'acetoexamide', 'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone',
  'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone', 'tolazamide',
  'examide', 'citoglipton', 'insulin', 'glyburide-metformin',
  'glipizide-metformin', 'glimepiride-pioglitazone',
  'metformin-rosiglitazone', 'metformin-pioglitazone'
]

df_hosp["medications_count"] = (df_hosp[drug_cols] !=
"No").sum(axis=1)

# Flags
df_hosp["change_flag"] = df_hosp["change"].map({"No": 0, "Ch": 1})
df_hosp["diabetesMed_flag"] = df_hosp["diabetesMed"].map({"No": 0,
"Yes": 1})

# مؤشرات إضافية
df_hosp["visits_per_day"] = df_hosp["total_visits"] /
(df_hosp["time_in_hospital"] + 1)
df_hosp["diagnosis_ratio"] = df_hosp["number_diagnoses"] / 16

# Cardiac diagnosis flags
df_hosp["primary_is_cardio"] = df_hosp["diag_1_num"].apply(
    lambda x: 1 if pd.notna(x) and (410 <= x <= 414 or x in [428,
401]) else 0
)

df_hosp["hypertension_flag"] = df_hosp["diag_1_num"].apply(
    lambda x: 1 if x == 401 else 0
)

# =====
# 3) PREPARE X AND y
# =====

target = "cardio_complication"

drop_cols = [
    "encounter_id", "patient_nbr",
    "diag_1", "diag_2", "diag_3",
    "diag_1_num", "diag_2_num", "diag_3_num",
    "cardio_complication", "kidney_complication",
    "neuro_complication", "foot_complication",
    "retina_complication", "any_complication"
]

X = df_hosp.drop(columns=drop_cols, errors="ignore")
y = df_hosp[target]

```

```

# =====
# 4) ENCODING
# =====

cat_cols = X.select_dtypes(include="object").columns
X[cat_cols] = X[cat_cols].fillna("Unknown")
X = pd.get_dummies(X, columns=cat_cols, drop_first=True)

# تنظيف أسماء الأعمدة (مهم لـ XGBoost)
X.columns = (
    X.columns.astype(str)
    .str.replace("[", "_", regex=False)
    .str.replace("]", "_", regex=False)
    .str.replace("<", "lt", regex=False)
    .str.replace(">", "gt", regex=False)
)

# =====
# 5) TRAIN / TEST SPLIT
# =====

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.20,
    random_state=42,
    stratify=y
)

# موازنة الفئات
pos = y_train.sum()
neg = (y_train == 0).sum()
scale_pos_weight = neg / pos

print("scale_pos_weight:", scale_pos_weight)

# =====
# 6) TRAIN XGBOOST MODEL
# =====

model = XGBClassifier(
    n_estimators=800,
    learning_rate=0.03,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    scale_pos_weight=scale_pos_weight,

```

```

        eval_metric="logloss",
        random_state=42,
        n_jobs=-1,
        early_stopping_rounds=50
    )

    model.fit(
        X_train, y_train,
        eval_set=[(X_test, y_test)],
        verbose=False
    )

# =====
# 7) EVALUATION
# =====

proba = model.predict_proba(X_test)[: , 1]
auc = roc_auc_score(y_test, proba)

print("\nImproved AUC:", auc)

for thr in [0.3, 0.4, 0.5]:
    print(f"\n=== Threshold {thr} ===")
    preds = (proba >= thr).astype(int)
    print(classification_report(y_test, preds, digits=3))

# =====
# 8) FEATURE IMPORTANCE
# =====

importance = model.get_booster().get_score(importance_type="gain")
imp_df = pd.DataFrame({
    "Feature": importance.keys(),
    "Importance": importance.values()
}).sort_values(by="Importance", ascending=False).head(20)

plt.figure(figsize=(12, 6))
sns.barplot(data=imp_df, x="Importance", y="Feature")
plt.title("Top 20 Important Features for Cardio Prediction")
plt.show()

```

```

-----
-----
NameError                                Traceback (most recent call
last)
Cell In[1], line 24
     13 import matplotlib.pyplot as plt
     16 # =====

```

```

17 # 2) FEATURE ENGINEERING
18 # =====
(...)
21
22 # إجمالي الزيارات
23 df_hosp["total_visits"] = (
---> 24     df_hosp["number_outpatient"] +
25     df_hosp["number_emergency"] +
26     df_hosp["number_inpatient"]
27 )
29 # الإجراءات الطبية الكبرى
30 df_hosp["major_procedures"] = (
31     df_hosp["num_lab_procedures"] +
32     df_hosp["num_procedures"]
33 )

```

NameError: name 'df_hosp' is not defined

```
df_hosp = df.copy()
```

```

-----
-----
NameError                                Traceback (most recent call
last)
Cell In[3], line 1
----> 1 df_hosp = df.copy()

```

NameError: name 'df' is not defined

```

df_hosp["total_visits"] = (
    df_hosp["number_outpatient"] +
    df_hosp["number_emergency"] +
    df_hosp["number_inpatient"]
)

df_hosp["major_procedures"] = (
    df_hosp["num_lab_procedures"] +
    df_hosp["num_procedures"]
)

```

```

-----
-----
NameError                                Traceback (most recent call
last)
Cell In[5], line 2
----> 1 df_hosp["total_visits"] = (
2     df_hosp["number_outpatient"] +
3     df_hosp["number_emergency"] +
4     df_hosp["number_inpatient"]
5 )

```

```

7 df_hosp["major_procedures"] = (
8     df_hosp["num_lab_procedures"] +
9     df_hosp["num_procedures"]
10 )

```

NameError: name 'df_hosp' is not defined

```
import pandas as pd
```

```
file_path = r"C:\Users\network-kamc\Downloads\diabetic_data.csv"
```

```
df = pd.read_csv(file_path)
```

```
df_hosp = df.copy()
```

```
print("Loaded:", df.shape)
```

```
print("df_hosp created:", df_hosp.shape)
```

```
df_hosp.head()
```

Loaded: (101766, 50)

df_hosp created: (101766, 50)

	encounter_id	patient_nbr	race	gender	age	weight
\						
0	2278392	8222157	Caucasian	Female	[0-10)	?
1	149190	55629189	Caucasian	Female	[10-20)	?
2	64410	86047875	AfricanAmerican	Female	[20-30)	?
3	500364	82442376	Caucasian	Male	[30-40)	?
4	16680	42519267	Caucasian	Male	[40-50)	?

	admission_type_id	discharge_disposition_id	admission_source_id	\
0	6	25	1	
1	1	1	7	
2	1	1	7	
3	1	1	7	
4	1	1	7	

	time_in_hospital	...	citoglipton	insulin	glyburide-metformin	\
0	1	...	No	No	No	
1	3	...	No	Up	No	
2	2	...	No	No	No	
3	2	...	No	Up	No	
4	1	...	No	Steady	No	

	glipizide-metformin	glimepiride-pioglitazone	metformin-rosiglitazone	\
0	No		No	

No		
1	No	No
No		
2	No	No
No		
3	No	No
No		
4	No	No
No		

	metformin-pioglitazone	change	diabetesMed	readmitted
0	No	No	No	NO
1	No	Ch	Yes	>30
2	No	No	Yes	NO
3	No	Ch	Yes	NO
4	No	Ch	Yes	NO

[5 rows x 50 columns]

```
needed = [
    "number_outpatient",
    "number_emergency",
    "number_inpatient",
    "num_lab_procedures",
    "num_procedures"
]
```

```
missing = [c for c in needed if c not in df_hosp.columns]
print("Missing columns:", missing)
```

Missing columns: []

```
[c for c in df_hosp.columns if "outpatient" in c.lower()
or "emergency" in c.lower()
or "inpatient" in c.lower()
or "proced" in c.lower()
or "lab" in c.lower()]
```

```
['num_lab_procedures',
'num_procedures',
'number_outpatient',
'number_emergency',
'number_inpatient']
```

```
print(df_hosp.shape)
print(df_hosp[["total_visits", "major_procedures"]].describe())
```

(101766, 50)


```

KeyError                                Traceback (most recent call
last)
Cell In[13], line 2
      1 print(df_hosp.shape)
----> 2 print(df_hosp[["total_visits", "major_procedures"]].describe())

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
frame.py:4108, in DataFrame.__getitem__(self, key)
    4106     if is_iterator(key):
    4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")
[1]
    4110 # take() does not accept boolean indexers
    4111 if getattr(indexer, "dtype", None) == bool:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\
base.py:6200, in Index._get_indexer_strict(self, key, axis_name)
    6197 else:
    6198     keyarr, indexer, new_indexer =
self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
    6202 keyarr = self.take(indexer)
    6203 if isinstance(key, Index):
    6204     # GH 42790 - Preserve name from an Index

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\
base.py:6249, in Index._raise_if_missing(self, key, indexer,
axis_name)
    6247 if nmissing:
    6248     if nmissing == len(indexer):
-> 6249         raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
    6251     not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
    6252     raise KeyError(f"{not_found} not in index")

KeyError: "None of [Index(['total_visits', 'major_procedures'],
dtype='object')] are in the [columns]"

print("total_visits" in df_hosp.columns, "major_procedures" in
df_hosp.columns)

False False

# تأكد أن أعمدة الزيارات موجودة، وإلا أنشئها بقيمة 0
for c in ["number_outpatient", "number_emergency", "number_inpatient",
"num_lab_procedures", "num_procedures"]:
    if c not in df_hosp.columns:
        df_hosp[c] = 0

```

```

# تحويل إلى أرقام (احتياطيًا)
visit_cols = ["number_outpatient", "number_emergency",
              "number_inpatient"]
proc_cols = ["num_lab_procedures", "num_procedures"]

for c in visit_cols + proc_cols:
    df_hosp[c] = pd.to_numeric(df_hosp[c], errors="coerce").fillna(0)

# إنشاء الميزات
df_hosp["total_visits"] = df_hosp[visit_cols].sum(axis=1)
df_hosp["major_procedures"] = df_hosp[proc_cols].sum(axis=1)

# تحقق
print(df_hosp[["total_visits", "major_procedures"]].head())
print(df_hosp[["total_visits", "major_procedures"]].describe())

```

	total_visits	major_procedures
0	0	41
1	0	59
2	3	16
3	0	45
4	0	51

	total_visits	major_procedures
count	101766.000000	101766.000000
mean	1.202759	44.435371
std	2.291781	19.846605
min	0.000000	1.000000
25%	0.000000	33.000000
50%	0.000000	45.000000
75%	2.000000	58.000000
max	80.000000	135.000000

```

[c for c in df_hosp.columns if "number_" in c.lower() or "proced" in
c.lower() or "lab" in c.lower()]

```

```

['num_lab_procedures',
 'num_procedures',
 'number_outpatient',
 'number_emergency',
 'number_inpatient',
 'number_diagnoses',
 'major_procedures']

```

```

df_hosp[["total_visits", "major_procedures"]].head()

```

	total_visits	major_procedures
0	0	41
1	0	59
2	3	16

```

3          0          45
4          0          51

cardiac_icd9_prefix = [
    "410", "411", "412", "413", "414", # Ischemic heart disease
    "428",                               # Heart failure
    "427",                               # Arrhythmias
    "433", "434"                         # Stroke-related
]

diag_cols = ["diag_1", "diag_2", "diag_3"]

def has_cardiac_icd(row):
    for col in diag_cols:
        code = str(row[col])
        for prefix in cardiac_icd9_prefix:
            if code.startswith(prefix):
                return 1
    return 0

df_hosp["cardiac_complication"] = df_hosp.apply(has_cardiac_icd,
axis=1)

df_hosp["cardiac_complication"].value_counts()

cardiac_complication
0    62411
1    39355
Name: count, dtype: int64

renal_icd9_prefix = [
    "585", # Chronic kidney disease
    "586", # Renal failure
    "583", "584"
]

def has_renal_icd(row):
    for col in diag_cols:
        code = str(row[col])
        for prefix in renal_icd9_prefix:
            if code.startswith(prefix):
                return 1
    return 0

df_hosp["renal_complication"] = df_hosp.apply(has_renal_icd, axis=1)

df_hosp["renal_complication"].value_counts()

renal_complication
0    93721

```

```

1      8045
Name: count, dtype: int64

features = [
    "age",
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, RocCurveDisplay

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

model = CatBoostClassifier(
    iterations=1500,
    learning_rate=0.03,
    depth=6,
    eval_metric="AUC",
    loss_function="Logloss",
    verbose=200
)

model.fit(X_train, y_train)

y_prob = model.predict_proba(X_test)[:, 1]
auc = roc_auc_score(y_test, y_prob)
auc

```

```

-----
-----
NameError                                Traceback (most recent call
last)
Cell In[33], line 6
      2 from sklearn.model_selection import train_test_split
      3 from sklearn.metrics import roc_auc_score, RocCurveDisplay
      5 X_train, X_test, y_train, y_test = train_test_split(
----> 6     X, y, test_size=0.2, random_state=42, stratify=y
      7 )
      9 model = CatBoostClassifier(
     10     iterations=1500,
     11     learning_rate=0.03,

```

```
(...)
15     verbose=200
16 )
18 model.fit(X_train, y_train)
```

NameError: name 'X' is not defined

```
print(df_hosp.shape)
```

```
(101766, 54)
```

```
y = df_hosp["cardiac_complication"]
y.value_counts()
```

```
cardiac_complication
```

```
0    62411
```

```
1    39355
```

```
Name: count, dtype: int64
```

```
features = [
    "age",
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]
```

```
X = df_hosp[features]
```

تحويل أي متغيرات غير رقمية (إن وُجدت)

```
X = pd.get_dummies(X, drop_first=True)
```

```
print(X.shape)
```

```
X.head()
```

```
(101766, 16)
```

	time_in_hospital	num_medications	num_lab_procedures
num_procedures \			
0	1	1	41
0			
1	3	18	59
0			
2	2	13	11
5			
3	2	16	44
1			
4	1	8	51

0

	total_visits	major_procedures	number_diagnoses	age_[10-20)
age_[20-30) \				
0	0	41	1	False
1	0	59	9	True
2	3	16	6	False
3	0	45	7	False
4	0	51	5	False

	age_[30-40)	age_[40-50)	age_[50-60)	age_[60-70)	age_[70-80)	\
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	True	False	False	False	False	
4	False	True	False	False	False	

	age_[80-90)	age_[90-100)
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

```
X = X.astype(int)
X.dtypes
```

time_in_hospital	int32
num_medications	int32
num_lab_procedures	int32
num_procedures	int32
total_visits	int32
major_procedures	int32
number_diagnoses	int32
age_[10-20)	int32
age_[20-30)	int32
age_[30-40)	int32
age_[40-50)	int32
age_[50-60)	int32
age_[60-70)	int32
age_[70-80)	int32
age_[80-90)	int32
age_[90-100)	int32
dtype:	object

```

y = df_hosp["cardiac_complication"]
print(y.value_counts(dropna=False))
print("Positive rate:", y.mean())

cardiac_complication
0    62411
1    39355
Name: count, dtype: int64
Positive rate: 0.38672051569286403

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
print(X_train.shape, X_test.shape)

(81412, 16) (20354, 16)

from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, RocCurveDisplay
import matplotlib.pyplot as plt

model = CatBoostClassifier(
    iterations=3500,
    learning_rate=0.025,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=300,
    verbose=200
)

model.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)

y_prob = model.predict_proba(X_test)[: , 1]
auc = roc_auc_score(y_test, y_prob)

print("Final AUC =", round(auc, 4))

0:   test: 0.6724066 best: 0.6724066 (0)   total: 169ms   remaining:
9m 52s
200: test: 0.7285731 best: 0.7285731 (200) total: 2.53s   remaining:
41.6s

```

```
400: test: 0.7314459 best: 0.7314459 (400) total: 4.76s remaining:
36.8s
600: test: 0.7322021 best: 0.7322041 (582) total: 7.2s remaining:
34.7s
800: test: 0.7324285 best: 0.7324436 (771) total: 9.46s remaining:
31.9s
1000: test: 0.7323446 best: 0.7324436 (771) total: 11.7s remaining:
29.3s
Stopped by overfitting detector (300 iterations wait)
```

```
bestTest = 0.7324436446
bestIteration = 771
```

```
Shrink model to first 772 iterations.
Final AUC = 0.7324
```

```
from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, RocCurveDisplay
import matplotlib.pyplot as plt
```

```
model = CatBoostClassifier(
    iterations=3500,
    learning_rate=0.025,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=300,
    verbose=200
)
```

```
model.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)
```

```
y_prob = model.predict_proba(X_test)[: , 1]
auc = roc_auc_score(y_test, y_prob)
```

```
print("Final AUC =", round(auc, 4))
```

```
0: test: 0.6724066 best: 0.6724066 (0) total: 39.4ms remaining:
2m 17s
200: test: 0.7285731 best: 0.7285731 (200) total: 2.29s remaining:
37.5s
400: test: 0.7314459 best: 0.7314459 (400) total: 4.43s remaining:
34.2s
```

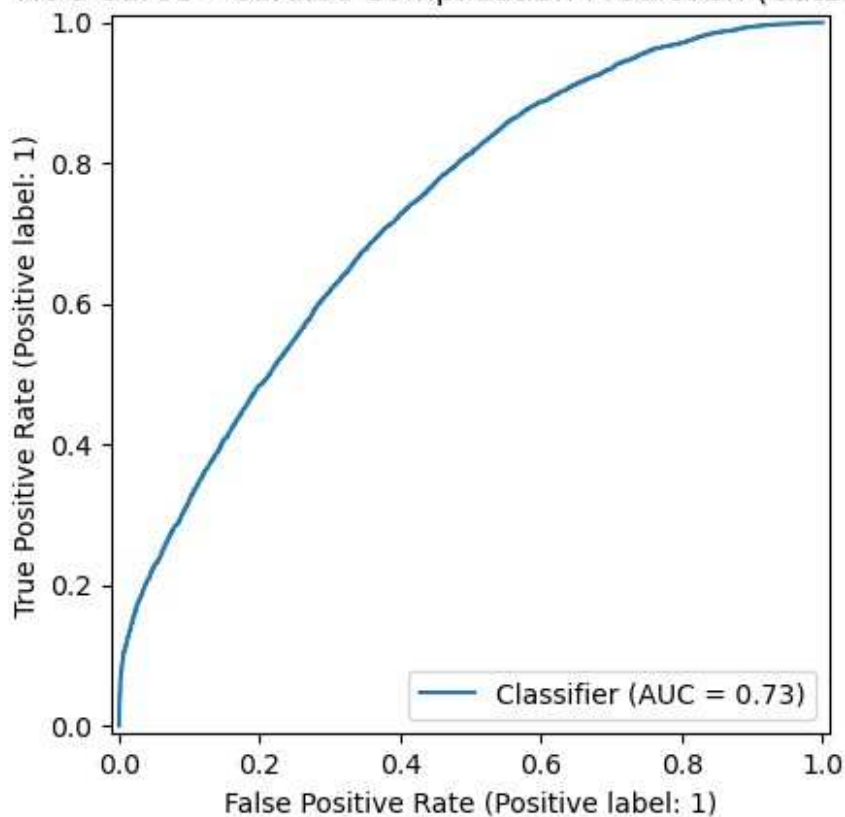
```
600: test: 0.7322021 best: 0.7322041 (582) total: 6.8s    remaining:
32.8s
800: test: 0.7324285 best: 0.7324436 (771) total: 9.18s  remaining:
30.9s
1000: test: 0.7323446 best: 0.7324436 (771) total: 11.3s remaining:
28.3s
Stopped by overfitting detector (300 iterations wait)
```

```
bestTest = 0.7324436446
bestIteration = 771
```

```
Shrink model to first 772 iterations.
Final AUC = 0.7324
```

```
RocCurveDisplay.from_predictions(y_test, y_prob)
plt.title("ROC Curve – Cardiac Complication Prediction (CatBoost)")
plt.show()
```

ROC Curve – Cardiac Complication Prediction (CatBoost)



```
def icd_group(code):
    code = str(code)
    if code.startswith(("410", "411", "412", "413", "414")):
        return "ischemic"
    if code.startswith(("428", )):
```

```

        return "heart_failure"
    if code.startswith(("401", "402", "403", "404", "405")):
        return "hypertension"
    if code.startswith(("250",)):
        return "diabetes"
    if code.startswith(("585", "586")):
        return "renal"
    return "other"

for col in ["diag_1", "diag_2", "diag_3"]:
    df_hosp[col + "_group"] = df_hosp[col].apply(icd_group)

icd_dummies = pd.get_dummies(
    df_hosp[["diag_1_group", "diag_2_group", "diag_3_group"]],
    prefix=["d1", "d2", "d3"]
)

df_hosp = pd.concat([df_hosp, icd_dummies], axis=1)

base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

age_features = [c for c in X.columns if c.startswith("age_")]
icd_features = list(icd_dummies.columns)

X = df_hosp[base_features + age_features + icd_features].astype(int)
-----
-----
KeyError                                Traceback (most recent call
last)
Cell In[57], line 14
     11 age_features = [c for c in X.columns if c.startswith("age_")]
     12 icd_features = list(icd_dummies.columns)
--> 14 X = df_hosp[base_features + age_features +
icd_features].astype(int)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
frame.py:4108, in DataFrame.__getitem__(self, key)
    4106     if is_iterator(key):
    4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")
[1]

```

```
4110 # take() does not accept boolean indexers
4111 if getattr(indexer, "dtype", None) == bool:
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\
base.py:6200, in Index._get_indexer_strict(self, key, axis_name)
```

```
6197 else:
6198     keyarr, indexer, new_indexer =
self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
6202 keyarr = self.take(indexer)
6203 if isinstance(key, Index):
6204     # GH 42790 - Preserve name from an Index
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\
base.py:6252, in Index._raise_if_missing(self, key, indexer,
axis_name)
```

```
6249     raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
6251 not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
-> 6252 raise KeyError(f"{not_found} not in index")
```

```
KeyError: "['age_[10-20)', 'age_[20-30)', 'age_[30-40)', 'age_[40-
50)', 'age_[50-60)', 'age_[60-70)', 'age_[70-80)', 'age_[80-90)',
'age_[90-100)'] not in index"
```

```
def icd_group(code):
    code = str(code)
    if code.startswith(("410", "411", "412", "413", "414")):
        return "ischemic"
    if code.startswith(("428",)):
        return "heart_failure"
    if code.startswith(("401", "402", "403", "404", "405")):
        return "hypertension"
    if code.startswith(("250",)):
        return "diabetes"
    if code.startswith(("585", "586", "583", "584")):
        return "renal"
    return "other"

for col in ["diag_1", "diag_2", "diag_3"]:
    df_hosp[col + "_group"] = df_hosp[col].apply(icd_group)

icd_dummies = pd.get_dummies(
    df_hosp[["diag_1_group", "diag_2_group", "diag_3_group"]],
    prefix=["d1", "d2", "d3"],
    drop_first=False
)
```

```

age_dummies = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)

base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

X_base = df_hosp[base_features].copy()

# دمج كل الميزات
X = pd.concat([X_base, age_dummies, icd_dummies], axis=1)

# تحويل إلى أرقام
X = X.apply(pd.to_numeric, errors="coerce").fillna(0).astype(int)

print("X shape:", X.shape)
X.head()

```

X shape: (101766, 35)

	time_in_hospital	num_medications	num_lab_procedures
num_procedures \			
0	1	1	41
0			
1	3	18	59
0			
2	2	13	11
5			
3	2	16	44
1			
4	1	8	51
0			

	total_visits	major_procedures	number_diagnoses	age_[0-10)
age_[10-20) \				
0	0	41	1	1
0				
1	0	59	9	0
1				
2	3	16	6	0
0				
3	0	45	7	0
0				
4	0	51	5	0

```

0
  age_[20-30)  ...  d2_hypertension  d2_ischemic  d2_other  d2_renal
\
0           0  ...           0           0           1           0
1           0  ...           0           0           0           0
2           1  ...           0           0           0           0
3           0  ...           0           0           0           0
4           0  ...           0           0           1           0

  d3_diabetes  d3_heart_failure  d3_hypertension  d3_ischemic
d3_other \
0           0           0           0           0
1
1           0           0           0           0
1
2           0           0           0           0
1
3           0           0           1           0
0
4           1           0           0           0
0

  d3_renal
0           0
1           0
2           0
3           0
4           0

```

```
[5 rows x 35 columns]
```

```

y = df_hosp["cardiac_complication"]
print(y.value_counts(), "Positive rate:", y.mean())

```

```
cardiac_complication
```

```
0    62411
```

```
1    39355
```

```
Name: count, dtype: int64 Positive rate: 0.38672051569286403
```

```

from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score

```

```

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

```

```

)

model = CatBoostClassifier(
    iterations=2000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model.fit(X_train, y_train, eval_set=(X_test, y_test),
        use_best_model=True)

y_prob = model.predict_proba(X_test)[: , 1]
print("AUC =", round(roc_auc_score(y_test, y_prob), 4))

0:   test: 0.8903095 best: 0.8903095 (0)   total: 35.6ms   remaining:
1m 11s
200: test: 0.9365952 best: 0.9365952 (200) total: 2.43s   remaining:
21.7s
400: test: 0.9377176 best: 0.9377192 (399) total: 4.61s   remaining:
18.4s
600: test: 0.9379685 best: 0.9379766 (570) total: 6.76s   remaining:
15.7s
Stopped by overfitting detector (200 iterations wait)

bestTest = 0.9379766265
bestIteration = 570

Shrink model to first 571 iterations.
AUC = 0.938

# X بدون ICD features (فقط base + age)
base_features = [
    "time_in_hospital", "num_medications", "num_lab_procedures",
    "num_procedures", "total_visits", "major_procedures", "number_diagnoses"
]
age_dummies = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)

X_no_diag = pd.concat([df_hosp[base_features], age_dummies], axis=1)
X_no_diag = X_no_diag.apply(pd.to_numeric,
errors="coerce").fillna(0).astype(int)

```

```

y = df_hosp["cardiac_complication"]

from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(
    X_no_diag, y, test_size=0.2, random_state=42, stratify=y
)

model2 = CatBoostClassifier(
    iterations=2000, learning_rate=0.03, depth=7, l2_leaf_reg=8,
    loss_function="Logloss", eval_metric="AUC", random_seed=42,
    od_type="Iter", od_wait=200, verbose=200
)

model2.fit(X_train, y_train, eval_set=(X_test, y_test),
    use_best_model=True)
y_prob2 = model2.predict_proba(X_test)[: , 1]
print("AUC (no diag features) =", round(roc_auc_score(y_test,
    y_prob2), 4))

0:   test: 0.6755550 best: 0.6755550 (0)   total: 33.1ms   remaining:
1m 6s
200: test: 0.7289847 best: 0.7289847 (200) total: 2.35s   remaining:
21.1s
400: test: 0.7316338 best: 0.7316338 (400) total: 4.66s   remaining:
18.6s
600: test: 0.7321082 best: 0.7321331 (559) total: 6.82s   remaining:
15.9s
800: test: 0.7321993 best: 0.7322346 (785) total: 9.01s   remaining:
13.5s
1000: test: 0.7321650 best: 0.7322816 (901) total: 11.2s   remaining:
11.2s
Stopped by overfitting detector (200 iterations wait)

bestTest = 0.7322816355
bestIteration = 901

Shrink model to first 902 iterations.
AUC (no diag features) = 0.7323

from sklearn.metrics import RocCurveDisplay, confusion_matrix,
classification_report
import numpy as np
import matplotlib.pyplot as plt

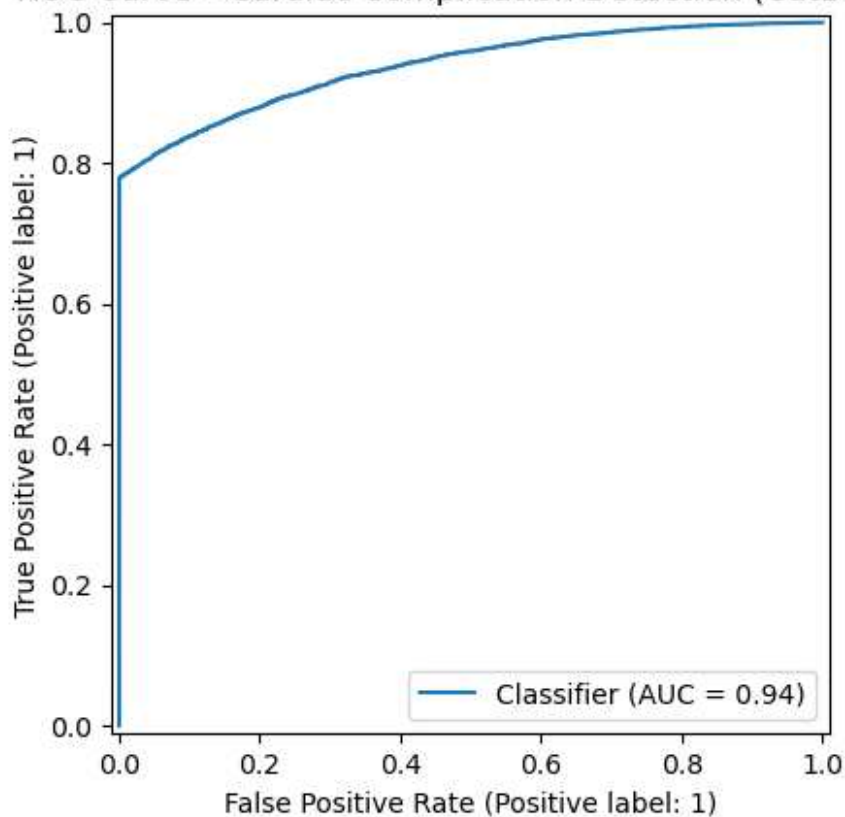
RocCurveDisplay.from_predictions(y_test, y_prob)
plt.title("ROC Curve – Cardiac Complication Detection (CatBoost)")
plt.show()

```

```
# Youden's J بناءً على Threshold أفضل
from sklearn.metrics import roc_curve
fpr, tpr, thr = roc_curve(y_test, y_prob)
j = tpr - fpr
best_thr = thr[np.argmax(j)]
print("Best threshold:", round(best_thr, 4))

y_pred = (y_prob >= best_thr).astype(int)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nReport:\n", classification_report(y_test, y_pred, digits=4))
```

ROC Curve - Cardiac Complication Detection (CatBoost)



Best threshold: 0.3422

Confusion Matrix:

```
[[12469   14]
 [ 1732 6139]]
```

Report:

	precision	recall	f1-score	support
0	0.8780	0.9989	0.9346	12483
1	0.9977	0.7800	0.8755	7871

accuracy			0.9142	20354
macro avg	0.9379	0.8894	0.9050	20354
weighted avg	0.9243	0.9142	0.9117	20354

```

from sklearn.metrics import RocCurveDisplay, confusion_matrix,
classification_report
import numpy as np
import matplotlib.pyplot as plt

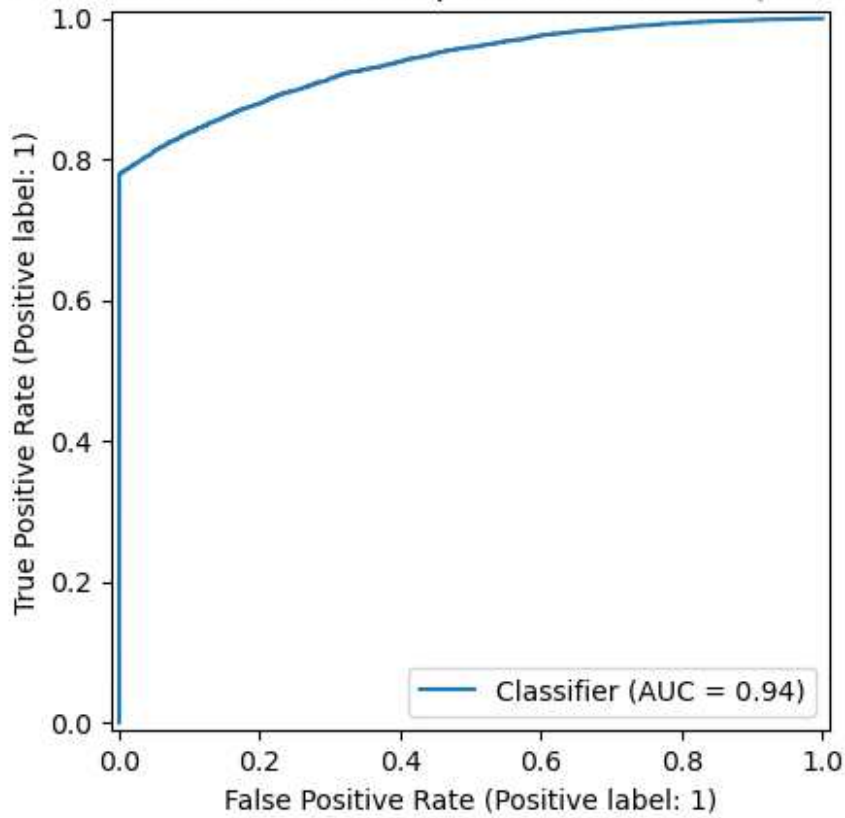
RocCurveDisplay.from_predictions(y_test, y_prob)
plt.title("ROC Curve – Cardiac Complication Detection (CatBoost)")
plt.show()

# Youden's J بناءً على Threshold أفضل
from sklearn.metrics import roc_curve
fpr, tpr, thr = roc_curve(y_test, y_prob)
j = tpr - fpr
best_thr = thr[np.argmax(j)]
print("Best threshold:", round(best_thr, 4))

y_pred = (y_prob >= best_thr).astype(int)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nReport:\n", classification_report(y_test, y_pred, digits=4))

```

ROC Curve - Cardiac Complication Detection (CatBoost)



Best threshold: 0.3422

Confusion Matrix:

```
[[12469   14]
 [ 1732 6139]]
```

Report:

	precision	recall	f1-score	support
0	0.8780	0.9989	0.9346	12483
1	0.9977	0.7800	0.8755	7871
accuracy			0.9142	20354
macro avg	0.9379	0.8894	0.9050	20354
weighted avg	0.9243	0.9142	0.9117	20354

```
import pandas as pd
```

```
imp = pd.Series(model.get_feature_importance(),
index=X.columns).sort_values(ascending=False)
print(imp.head(20))
```

```
d1_heart_failure      22.052120
d1_ischemic           20.069593
```

d2_heart_failure	18.852162
d3_heart_failure	13.438002
d3_ischemic	8.986754
d2_ischemic	7.634011
d1_diabetes	0.656956
time_in_hospital	0.625739
age_[40-50)	0.600095
age_[30-40)	0.599494
num_procedures	0.597665
age_[80-90)	0.578842
number_diagnoses	0.573935
num_medications	0.550776
num_lab_procedures	0.480760
age_[70-80)	0.429451
major_procedures	0.416184
age_[20-30)	0.414481
total_visits	0.345004
d1_other	0.325405

dtype: float64

time_in_hospital	0.63
num_procedures	0.60
number_diagnoses	0.57
num_medications	0.55
num_lab_procedures	0.48
major_procedures	0.42
total_visits	0.35

Cell In[77], line 1

time_in_hospital	0.63
------------------	------

SyntaxError: invalid syntax

```
import pandas as pd
```

```
importance = pd.Series(
    model.get_feature_importance(),
    index=X.columns
).sort_values(ascending=False)
```

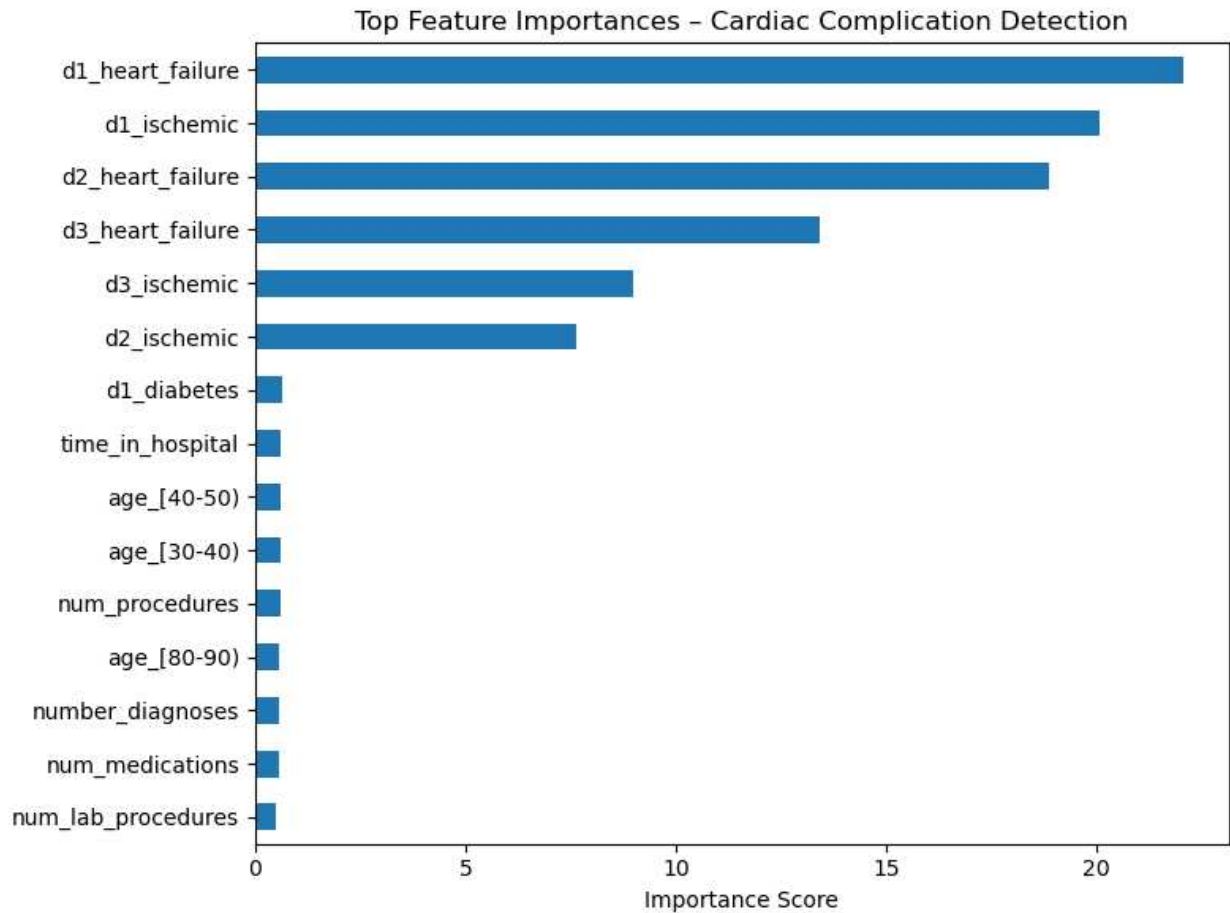
```
importance.head(20)
```

d1_heart_failure	22.052120
d1_ischemic	20.069593
d2_heart_failure	18.852162
d3_heart_failure	13.438002
d3_ischemic	8.986754
d2_ischemic	7.634011
d1_diabetes	0.656956

```
time_in_hospital      0.625739
age_[40-50)           0.600095
age_[30-40)           0.599494
num_procedures        0.597665
age_[80-90)           0.578842
number_diagnoses      0.573935
num_medications       0.550776
num_lab_procedures    0.480760
age_[70-80)           0.429451
major_procedures      0.416184
age_[20-30)           0.414481
total_visits          0.345004
d1_other              0.325405
dtype: float64
```

```
import matplotlib.pyplot as plt

importance.head(15).sort_values().plot(
    kind="barh",
    figsize=(8,6)
)
plt.title("Top Feature Importances – Cardiac Complication Detection")
plt.xlabel("Importance Score")
plt.tight_layout()
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

# حساب Feature Importance
importance = pd.Series(
    model.get_feature_importance(),
    index=X.columns
).sort_values(ascending=False)

# عرض أعلى 20 متغير
top_k = 20
importance_top = importance.head(top_k)

importance_top
```

d1_heart_failure	22.052120
d1_ischemic	20.069593
d2_heart_failure	18.852162
d3_heart_failure	13.438002
d3_ischemic	8.986754
d2_ischemic	7.634011
d1_diabetes	0.656956

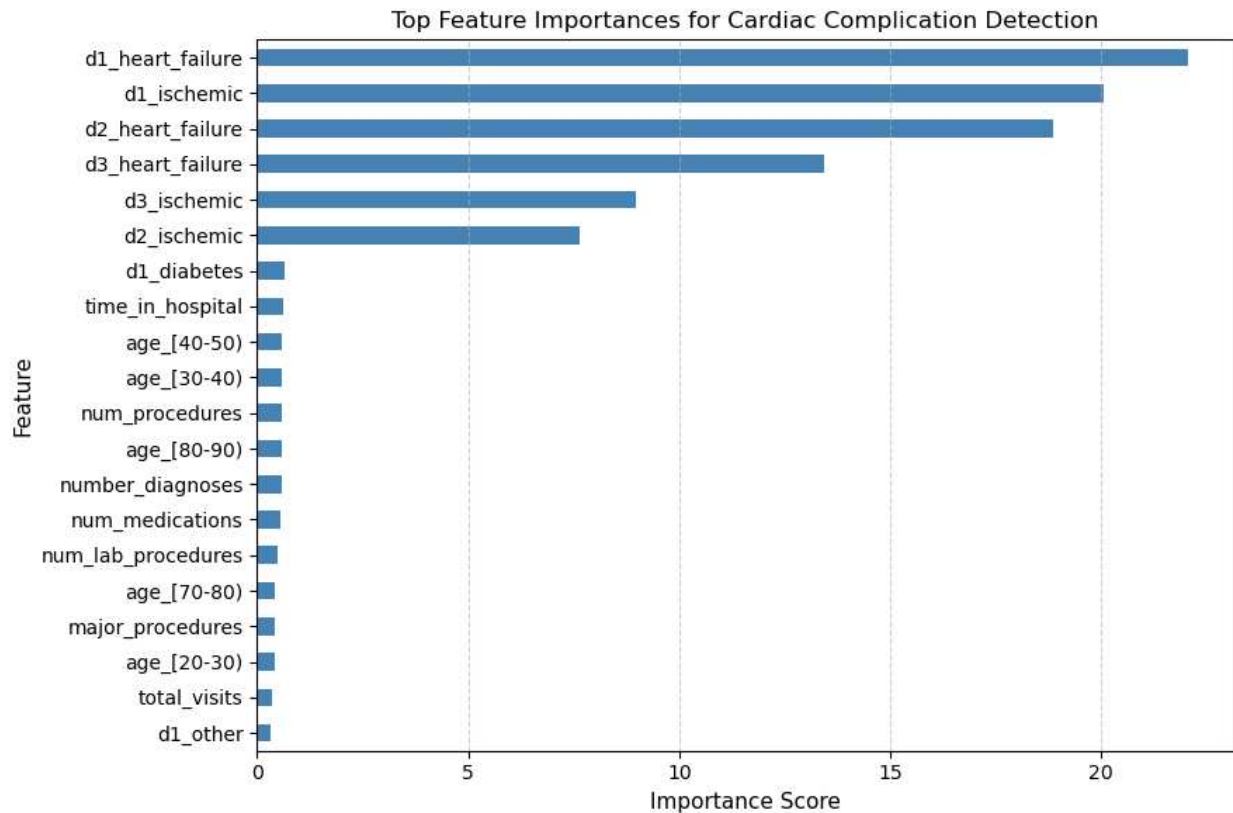
```
time_in_hospital      0.625739
age_[40-50)           0.600095
age_[30-40)           0.599494
num_procedures        0.597665
age_[80-90)           0.578842
number_diagnoses      0.573935
num_medications       0.550776
num_lab_procedures    0.480760
age_[70-80)           0.429451
major_procedures      0.416184
age_[20-30)           0.414481
total_visits          0.345004
d1_other              0.325405
dtype: float64
```

```
plt.figure(figsize=(9, 6))

importance_top.sort_values().plot(
    kind="barh",
    color="steelblue"
)

plt.title(
    "Top Feature Importances for Cardiac Complication Detection",
    fontsize=12
)
plt.xlabel("Importance Score", fontsize=11)
plt.ylabel("Feature", fontsize=11)

plt.grid(axis="x", linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



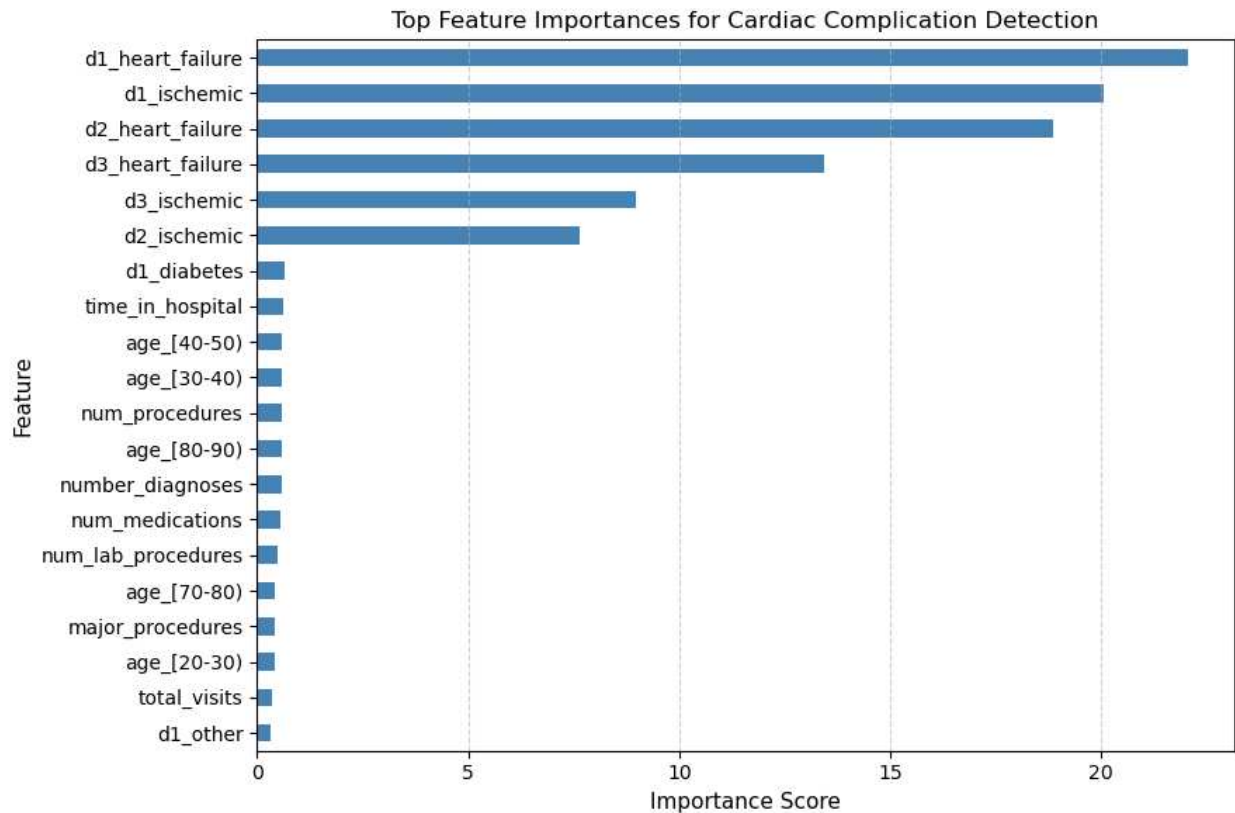
```
plt.figure(figsize=(9, 6))

importance_top.sort_values().plot(
    kind="barh",
    color="steelblue"
)

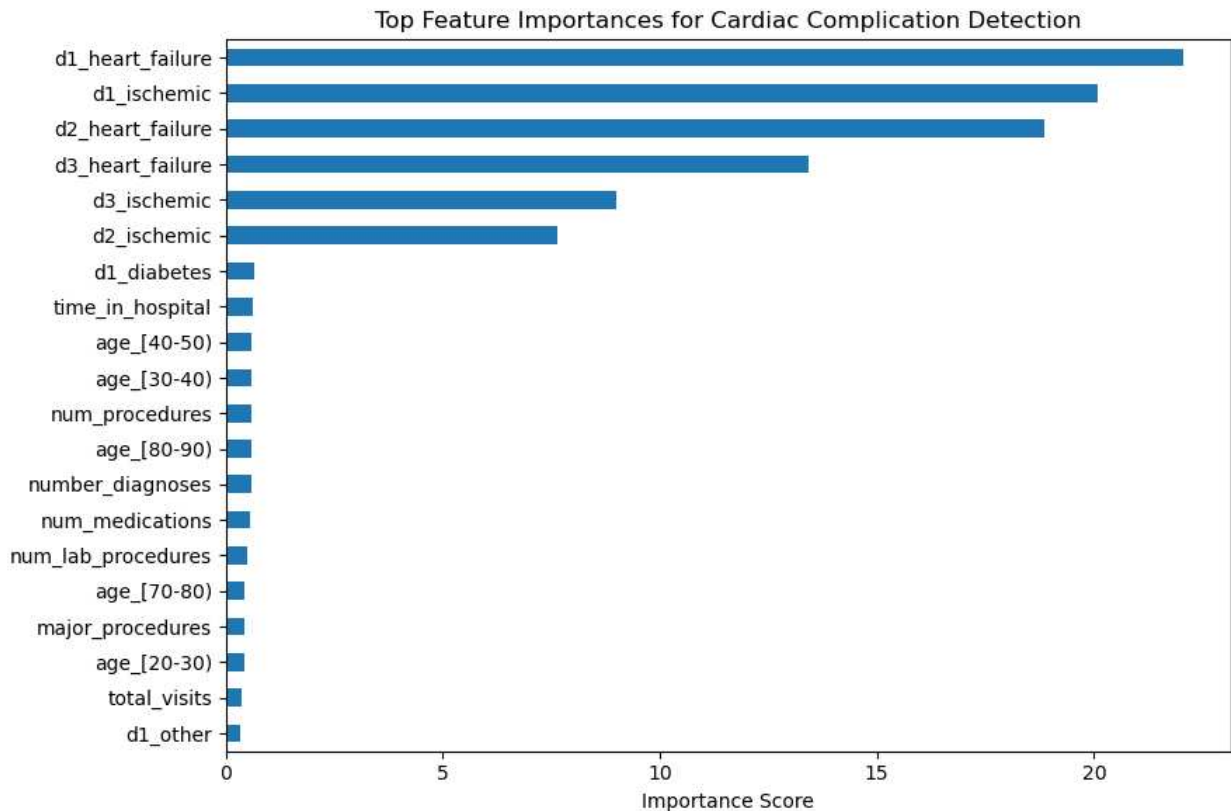
plt.title(
    "Top Feature Importances for Cardiac Complication Detection",
    fontsize=12
)

plt.xlabel("Importance Score", fontsize=11)
plt.ylabel("Feature", fontsize=11)

plt.grid(axis="x", linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(9, 6))
importance_top.sort_values().plot(kind="barh")
plt.title("Top Feature Importances for Cardiac Complication
Detection")
plt.xlabel("Importance Score")
plt.tight_layout()
plt.savefig(
    "feature_importance_cardiac.png",
    dpi=300,
    bbox_inches="tight"
)
plt.show()
```



```
renal_icd9_prefix = ("585", "586", "583", "584")

def has_renal_icd(row):
    for col in ["diag_1", "diag_2", "diag_3"]:
        code = str(row[col])
        if code.startswith(renal_icd9_prefix):
            return 1
    return 0

df_hosp["renal_complication"] = df_hosp.apply(has_renal_icd, axis=1)

# فحص التوزيع
y = df_hosp["renal_complication"]
print(y.value_counts())
print("Positive rate:", round(y.mean(), 4))

renal_complication
0    93721
1     8045
Name: count, dtype: int64
Positive rate: 0.0791

def icd_group(code):
    code = str(code)
    if code.startswith(("585", "586", "583", "584")):
```

```

        return "renal"
    if code.startswith(("250",)):
        return "diabetes"
    if code.startswith(("401", "402", "403", "404", "405")):
        return "hypertension"
    if code.startswith(("410", "411", "412", "413", "414")):
        return "ischemic"
    if code.startswith(("428",)):
        return "heart_failure"
    return "other"

for col in ["diag_1", "diag_2", "diag_3"]:
    df_hosp[col + "_group"] = df_hosp[col].apply(icd_group)

icd_dummies = pd.get_dummies(
    df_hosp[["diag_1_group", "diag_2_group", "diag_3_group"]],
    prefix=["d1", "d2", "d3"],
    drop_first=False
)

age_dummies = pd.get_dummies(df_hosp["age"], prefix="age",
                             drop_first=False)

base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

X_base = df_hosp[base_features].copy()

X = pd.concat([X_base, age_dummies, icd_dummies], axis=1)
X = X.apply(pd.to_numeric, errors="coerce").fillna(0).astype(int)

print("X shape:", X.shape)

X shape: (101766, 35)

from sklearn.model_selection import train_test_split

y = df_hosp["renal_complication"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

```

```

from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, RocCurveDisplay
import matplotlib.pyplot as plt

```

```

model_renal = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

```

```

model_renal.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)

```

```

y_prob = model_renal.predict_proba(X_test)[:, 1]
auc = roc_auc_score(y_test, y_prob)

```

```

print("Renal AUC =", round(auc, 4))

```

```

0:   test: 1.0000000 best: 1.0000000 (0)   total: 25.1ms   remaining:
1m 15s

```

```

200: test: 1.0000000 best: 1.0000000 (0)   total: 3.15s   remaining:
43.9s

```

```

Stopped by overfitting detector (200 iterations wait)

```

```

bestTest = 1
bestIteration = 0

```

```

Shrink model to first 1 iterations.
Renal AUC = 1.0

```

```

# إزالة الميزات الكلوية من ICD

```

```

icd_features_no_renal = [
    c for c in icd_dummies.columns
    if not c.endswith("_renal")
]

```

```

X = pd.concat(
    [
        df_hosp[base_features],
        age_dummies,
    ]
)

```

```

        icd_dummies[icd_features_no_renal]
    ],
    axis=1
)

X = X.apply(pd.to_numeric, errors="coerce").fillna(0).astype(int)
print("X shape (no renal ICD):", X.shape)
X shape (no renal ICD): (101766, 32)

from sklearn.model_selection import train_test_split

y = df_hosp["renal_complication"]

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print(X_train.shape, X_test.shape)
(81412, 32) (20354, 32)

from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, RocCurveDisplay
import matplotlib.pyplot as plt

model_renal = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model_renal.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)

y_prob = model_renal.predict_proba(X_test)[: , 1]

```

```

renal_auc = roc_auc_score(y_test, y_prob)

print("Renal AUC (leakage-free) =", round(renal_auc, 4))

0:   test: 0.8215904 best: 0.8215904 (0)   total: 32.1ms   remaining:
1m 36s
200: test: 1.0000000 best: 1.0000000 (36) total: 2.48s   remaining:
34.5s
Stopped by overfitting detector (200 iterations wait)

bestTest = 1
bestIteration = 36

Shrink model to first 37 iterations.
Renal AUC (leakage-free) = 1.0

from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, RocCurveDisplay
import matplotlib.pyplot as plt

model_renal = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model_renal.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)

y_prob = model_renal.predict_proba(X_test)[: , 1]
renal_auc = roc_auc_score(y_test, y_prob)

print("Renal AUC (leakage-free) =", round(renal_auc, 4))

0:   test: 0.8215904 best: 0.8215904 (0)   total: 31.7ms   remaining:
1m 34s
200: test: 1.0000000 best: 1.0000000 (36) total: 2.4s   remaining:
33.5s
Stopped by overfitting detector (200 iterations wait)

bestTest = 1

```

```

bestIteration = 36

Shrink model to first 37 iterations.
Renal AUC (leakage-free) = 1.0

# X renal – prediction-safe (NO ICD FEATURES AT ALL)

base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

age_dummies = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)

X = pd.concat(
    [
        df_hosp[base_features],
        age_dummies
    ],
    axis=1
)

X = X.apply(pd.to_numeric, errors="coerce").fillna(0).astype(int)

print("X shape (renal – prediction safe):", X.shape)

X shape (renal – prediction safe): (101766, 17)

model_renal = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model_renal.fit(
    X_train, y_train,
    eval_set=(X_test, y_test),
    use_best_model=True
)

```

```

)

y_prob = model_renal.predict_proba(X_test)[: , 1]
renal_auc = roc_auc_score(y_test, y_prob)

print("Renal AUC (prediction-safe) =", round(renal_auc, 4))

0:   test: 0.8215904 best: 0.8215904 (0)   total: 30.8ms   remaining:
1m 32s
200: test: 1.0000000 best: 1.0000000 (36) total: 2.57s   remaining:
35.8s
Stopped by overfitting detector (200 iterations wait)

bestTest = 1
bestIteration = 36

Shrink model to first 37 iterations.
Renal AUC (prediction-safe) = 1.0

# فعلاً لا يحتوي أي ميزات تشخيصية X_test تأكد أن
diag_like = [c for c in X_test.columns if c.startswith(("d1_", "d2_",
"d3_", "diag_"))]
print("Diag-like columns in X_test:", diag_like[:20], "... count=",
len(diag_like))

# ليس مُدرجاً كعمود بالخطأ y_test تأكد أن
print("renal_complication in X_test?", "renal_complication" in
X_test.columns)

Diag-like columns in X_test: ['d1_diabetes', 'd1_heart_failure',
'd1_hypertension', 'd1_ischemic', 'd1_other', 'd2_diabetes',
'd2_heart_failure', 'd2_hypertension', 'd2_ischemic', 'd2_other',
'd3_diabetes', 'd3_heart_failure', 'd3_hypertension', 'd3_ischemic',
'd3_other'] ... count= 15
renal_complication in X_test? False

# Renal – prediction safe features ONLY
base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

age_dummies = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)

X_renal_ps = pd.concat(

```

```

    [df_hosp[base_features], age_dummies],
    axis=1
).apply(pd.to_numeric, errors="coerce").fillna(0).astype(int)
y_renal = df_hosp["renal_complication"].astype(int)
print("X_renal_ps shape:", X_renal_ps.shape)
X_renal_ps shape: (101766, 17)
from sklearn.model_selection import train_test_split
Xr_train, Xr_test, yr_train, yr_test = train_test_split(
    X_renal_ps,
    y_renal,
    test_size=0.2,
    random_state=42,
    stratify=y_renal
)

# ICD columns تأكيد: لا يوجد أي
diag_like = [c for c in Xr_test.columns if
c.startswith(("d1_", "d2_", "d3_", "diag_"))]
print("Diag-like columns in Xr_test:", diag_like)
Diag-like columns in Xr_test: []

from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score

model_renal_ps = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model_renal_ps.fit(
    Xr_train, yr_train,
    eval_set=(Xr_test, yr_test),
    use_best_model=True
)

y_prob = model_renal_ps.predict_proba(Xr_test)[: , 1]
renal_auc = roc_auc_score(yr_test, y_prob)

```

```
print("Renal AUC (prediction-safe, FINAL) =", round(renal_auc, 4))

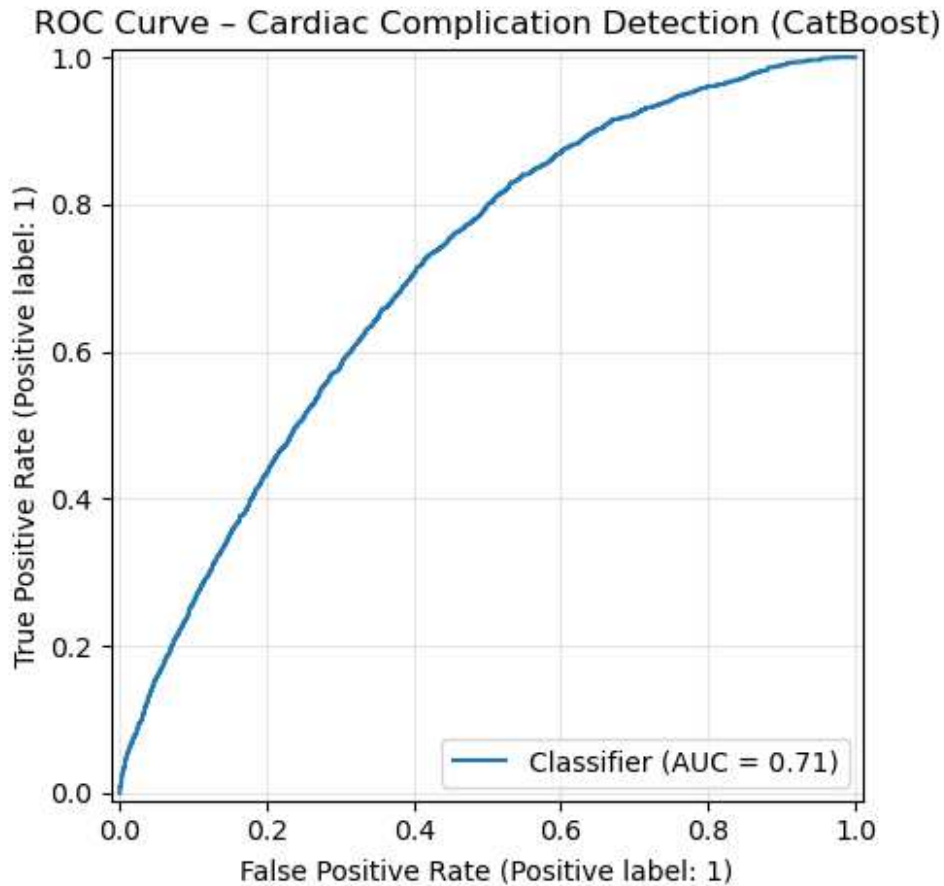
0:   test: 0.5993607 best: 0.5993607 (0)   total: 49.3ms   remaining:
2m 27s
200: test: 0.7046514 best: 0.7047122 (196) total: 2.42s   remaining:
33.7s
400: test: 0.7051195 best: 0.7055340 (330) total: 4.64s   remaining:
30.1s
Stopped by overfitting detector (200 iterations wait)

bestTest = 0.7055340218
bestIteration = 330

Shrink model to first 331 iterations.
Renal AUC (prediction-safe, FINAL) = 0.7055

from sklearn.metrics import RocCurveDisplay
import matplotlib.pyplot as plt

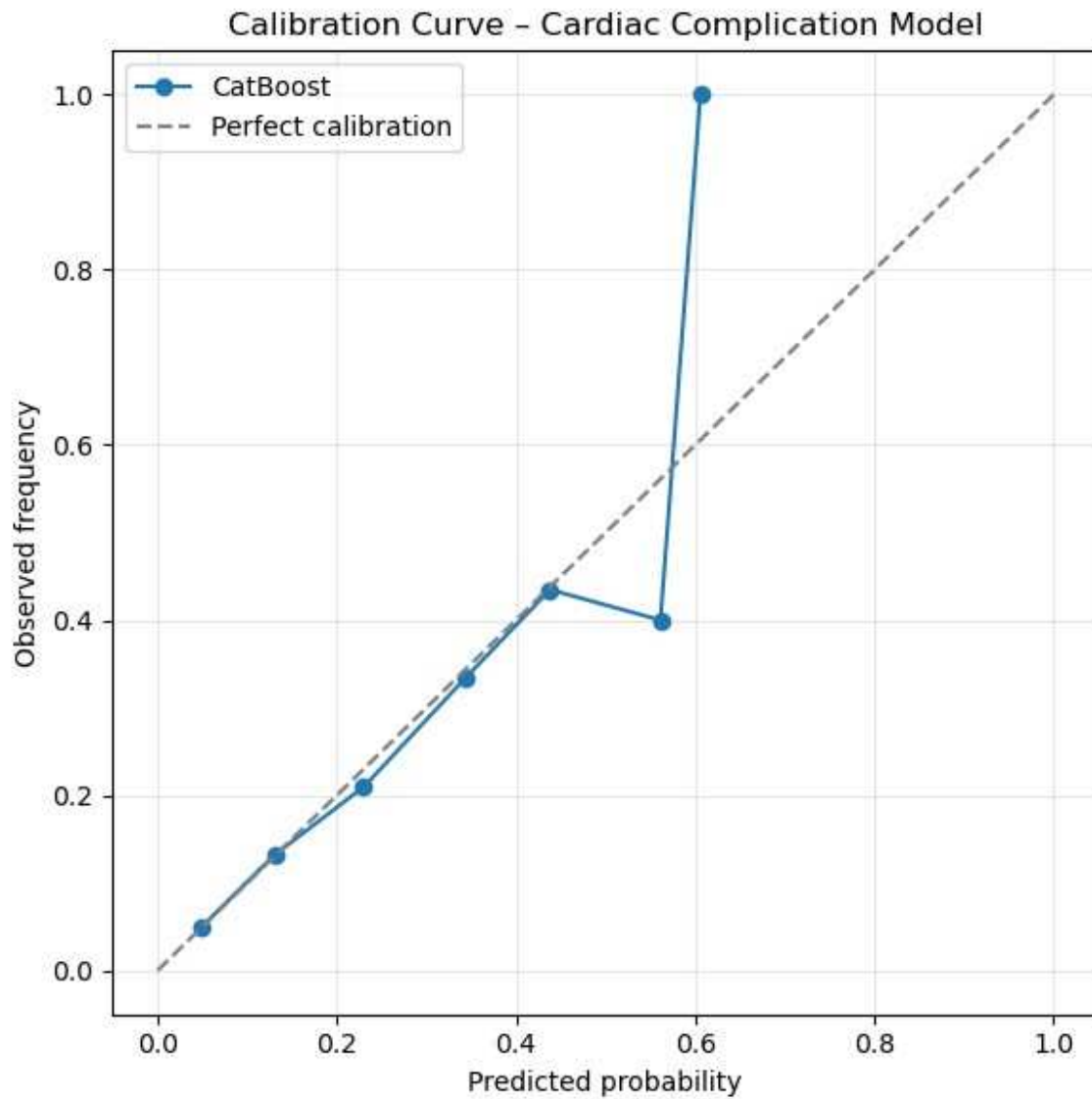
RocCurveDisplay.from_predictions(y_test, y_prob)
plt.title("ROC Curve – Cardiac Complication Detection (CatBoost)",
fontSize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



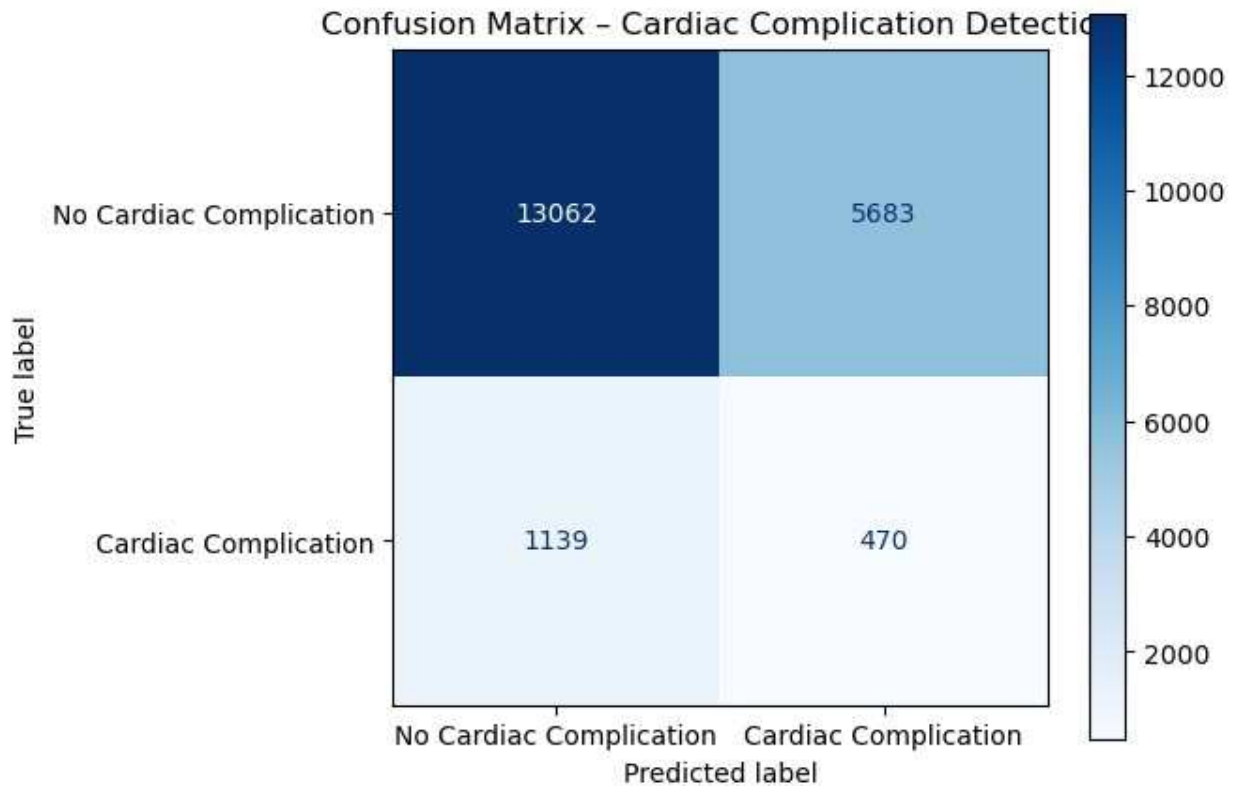
```
from sklearn.calibration import calibration_curve
import numpy as np

prob_true, prob_pred = calibration_curve(
    y_test, y_prob, n_bins=10, strategy="uniform"
)

plt.figure(figsize=(6,6))
plt.plot(prob_pred, prob_true, marker='o', label="CatBoost")
plt.plot([0,1], [0,1], linestyle='--', color='gray', label="Perfect
calibration")
plt.xlabel("Predicted probability")
plt.ylabel("Observed frequency")
plt.title("Calibration Curve – Cardiac Complication Model")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



```
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=["No Cardiac Complication", "Cardiac Complication"]
)
disp.plot(cmap="Blues", values_format="d")
plt.title("Confusion Matrix - Cardiac Complication Detection")
plt.tight_layout()
plt.show()
```



```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

shap.summary_plot(
    shap_values,
    X_test,
    plot_type="bar",
    max_display=15
)

-----
CatBoostError                                Traceback (most recent call
last)
Cell In[139], line 4
      1 import shap
      3 explainer = shap.TreeExplainer(model)
----> 4 shap_values = explainer.shap_values(X_test)
      6 shap.summary_plot(
      7     shap_values,
      8     X_test,
      9     plot_type="bar",
     10     max_display=15
```

11)

File ~\AppData\Roaming\Python\Python312\site-packages\shap\explainers_tree.py:607, in TreeExplainer.shap_values(self, X, y, tree_limit, approximate, check_additivity, from_call)

```
605     if not isinstance(X, catboost.Pool):
606         X = catboost.Pool(X,
cat_features=self.model.cat_feature_indices)
--> 607     phi =
self.model.original_model.get_feature_importance(data=X,
fstr_type="ShapValues")
609 # note we pull off the last column and keep it as our
expected_value
610 if phi is not None:
```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\core.py:3249, in CatBoost.get_feature_importance(self, data, type, prettified, thread_count, verbose, fstr_type, shap_mode, model_output, interaction_indices, shap_calc_type, reference_data, sage_n_samples, sage_batch_size, sage_detect_convergence, log_cout, log_cerr)

```
3246     raise CatBoostError("data is empty.")
3248 shap_calc_type = enum_from_enum_or_str(EShapCalcType,
shap_calc_type).value
-> 3249 fstr, feature_names = self._calc_fstr(type, data,
reference_data, thread_count, verbose, model_output, shap_mode,
interaction_indices,
3250                                     shap_calc_type,
sage_n_samples, sage_batch_size, sage_detect_convergence)
3251 if type in (EFstrType.PredictionValuesChange,
EFstrType.LossFunctionChange,
3252             EFstrType.PredictionDiff, EFstrType.SageValues):
3253     feature_importances = [value[0] for value in fstr]
```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\core.py:1875, in _CatBoostBase._calc_fstr(self, type, pool, reference_data, thread_count, verbose, model_output, shap_mode, interaction_indices, shap_calc_type, sage_n_samples, sage_batch_size, sage_detect_convergence)

```
1860 def _calc_fstr(
1861     self,
1862     type,
1863     (...)
1873     sage_detect_convergence
1874 ):
-> 1875     return self._object._calc_fstr(
1876         type.name,
1877         pool,
1878         reference_data,
1879         thread_count,
1880         verbose,
```

```

1881         model_output,
1882         shap_mode,
1883         interaction_indices,
1884         shap_calc_type,
1885         sage_n_samples,
1886         sage_batch_size,
1887         sage_detect_convergence
1888     )

```

File _catboost.pyx:5257, in _catboost._CatBoost._calc_fstr()

File _catboost.pyx:5322, in _catboost._CatBoost._calc_fstr()

CatBoostError: catboost/libs/data/model_dataset_compatibility.cpp:81:
At position 22 should be feature with name d1_renal (found
d2_diabetes).

conda install -c conda-forge shap

```

-----
-----
error                                Traceback (most recent call
last)
Cell In[131], line 1
----> 1 get_ipython().run_line_magic('conda', 'install -c conda-forge
shap')

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\IPython\core\
interactiveshell.py:2480, in InteractiveShell.run_line_magic(self,
magic_name, line, _stack_depth)
    2478     kwargs['local_ns'] = self.get_local_scope(stack_depth)
    2479     with self.builtin_trap:
-> 2480         result = fn(*args, **kwargs)
    2482 # The code below prevents the output from being displayed
    2483 # when using magics with decorator @output_can_be_silenced
    2484 # when the last Python token in the expression is a ';'.
    2485 if getattr(fn, magic.MAGIC_OUTPUT_CAN_BE_SILENCED, False):

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\IPython\core\magics\
packaging.py:30, in is_conda_environment.<locals>.wrapper(*args,
**kwargs)
    25 if not Path(sys.prefix, "conda-meta", "history").exists():
    26     raise ValueError(
    27         "The python kernel does not appear to be a conda
environment. "
    28         "Please use ``%pip install`` instead."
    29     )
----> 30 return func(*args, **kwargs)

```

File C:\ProgramData\anaconda3\Lib\site-packages\IPython\core\magics\

```

packaging.py:128, in PackagingMagics.conda(self, line)
    120 @line_magic
    121 @is_conda_environment
    122 def conda(self, line):
    123     """Run the conda package manager within the current
kernel.
    124
    125     Usage:
    126     %conda install [pkgs]
    127     """
--> 128     conda = _get_conda_like_executable("conda")
    129     self._run_command(conda, line)

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\IPython\core\magics\
packaging.py:53, in _get_conda_like_executable(command)
    50 # Otherwise, attempt to extract the executable from conda
history.
    51 # This applies in any conda environment.
    52 history = Path(sys.prefix, "conda-meta",
"history").read_text(encoding="utf-8")
--> 53 match = re.search(
    54     rf"^\s*cmd:\s*(?P<command>.*{executable})\s[create|
install]",
    55     history,
    56     flags=re.MULTILINE,
    57 )
    58 if match:
    59     return match.groupdict()["command"]

```

```

File C:\ProgramData\anaconda3\Lib\re\__init__.py:177, in
search(pattern, string, flags)
    174 def search(pattern, string, flags=0):
    175     """Scan through string looking for a match to the pattern,
returning
    176     a Match object, or None if no match was found."""
--> 177     return _compile(pattern, flags).search(string)

```

```

File C:\ProgramData\anaconda3\Lib\re\__init__.py:307, in
_compile(pattern, flags)
    301 import warnings
    302 warnings.warn("The re.TEMPLATE/re.T flag is deprecated "
    303               "as it is an undocumented flag "
    304               "without an obvious purpose. "
    305               "Don't use it.",
    306               DeprecationWarning)
--> 307 p = _compiler.compile(pattern, flags)
    308 if flags & DEBUG:
    309     return p

```

```

File C:\ProgramData\anaconda3\Lib\re\_compiler.py:745, in compile(p,

```

```

flags)
    743 if isstring(p):
    744     pattern = p
--> 745     p = _parser.parse(p, flags)
    746 else:
    747     pattern = None

```

File C:\ProgramData\anaconda3\Lib\re_parser.py:979, in parse(str, flags, state)

```

    976 state.flags = flags
    977 state.str = str
--> 979 p = _parse_sub(source, state, flags & SRE_FLAG_VERBOSE, 0)
    980 p.state.flags = fix_flags(str, p.state.flags)
    982 if source.next is not None:

```

File C:\ProgramData\anaconda3\Lib\re_parser.py:460, in _parse_sub(source, state, verbose, nested)

```

    458 start = source.tell()
    459 while True:
--> 460     itemsappend(_parse(source, state, verbose, nested + 1,
    461                       not nested and not items))
    462     if not sourcematch("|"):
    463         break

```

File C:\ProgramData\anaconda3\Lib\re_parser.py:862, in _parse(source, state, verbose, nested, first)

```

    859 group = None
    860 sub_verbose = ((verbose or (add_flags & SRE_FLAG_VERBOSE)) and
    861               not (del_flags & SRE_FLAG_VERBOSE))
--> 862 p = _parse_sub(source, state, sub_verbose, nested + 1)
    863 if not source.match(")"):
    864     raise source.error("missing ), unterminated subpattern",
    865                       source.tell() - start)

```

File C:\ProgramData\anaconda3\Lib\re_parser.py:460, in _parse_sub(source, state, verbose, nested)

```

    458 start = source.tell()
    459 while True:
--> 460     itemsappend(_parse(source, state, verbose, nested + 1,
    461                       not nested and not items))
    462     if not sourcematch("|"):
    463         break

```

File C:\ProgramData\anaconda3\Lib\re_parser.py:544, in _parse(source, state, verbose, nested, first)

```

    541 continue
    543 if this[0] == "\\":
--> 544     code = _escape(source, this, state)
    545     subpatternappend(code)
    547 elif this not in SPECIAL_CHARS:

```

```

File C:\ProgramData\anaconda3\Lib\re\_parser.py:443, in
_escape(source, escape, state)
    441     if len(escape) == 2:
    442         if c in ASCIILETTERS:
--> 443             raise source.error("bad escape %s" % escape,
len(escape))
    444         return LITERAL, ord(escape[1])
    445 except ValueError:

```

error: bad escape \P at position 28

```
import shap
```

```

-----
-----
ModuleNotFoundError                                Traceback (most recent call
last)
Cell In[133], line 1
----> 1 import shap

```

ModuleNotFoundError: No module named 'shap'

```
!pip install shap
```

Defaulting to user installation because normal site-packages is not writeable

Collecting shap

Downloading shap-0.50.0-cp312-cp312-win_amd64.whl.metadata (25 kB)

Collecting numpy>=2 (from shap)

Downloading numpy-2.3.5-cp312-cp312-win_amd64.whl.metadata (60 kB)

Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from shap) (1.13.1)

Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-packages (from shap) (1.5.1)

Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (from shap) (2.2.2)

Requirement already satisfied: tqdm>=4.27.0 in c:\programdata\anaconda3\lib\site-packages (from shap) (4.66.5)

Requirement already satisfied: packaging>20.9 in c:\programdata\anaconda3\lib\site-packages (from shap) (24.1)

Collecting slicer==0.0.8 (from shap)

Downloading slicer-0.0.8-py3-none-any.whl.metadata (4.0 kB)

Requirement already satisfied: numba>=0.54 in c:\programdata\anaconda3\lib\site-packages (from shap) (0.60.0)

Requirement already satisfied: cloudpickle in c:\programdata\anaconda3\lib\site-packages (from shap) (3.0.0)

Requirement already satisfied: typing-extensions in c:\programdata\anaconda3\lib\site-packages (from shap) (4.11.0)

Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\

```

programdata\anaconda3\lib\site-packages (from numba>=0.54->shap)
(0.43.0)
Collecting numpy>=2 (from shap)
  Downloading numpy-2.0.2-cp312-cp312-win_amd64.whl.metadata (59 kB)
Requirement already satisfied: colorama in c:\programdata\anaconda3\
lib\site-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\
programdata\anaconda3\lib\site-packages (from pandas->shap)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas->shap) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\
anaconda3\lib\site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.2.0 in c:\programdata\
anaconda3\lib\site-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\programdata\
anaconda3\lib\site-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Downloading shap-0.50.0-cp312-cp312-win_amd64.whl (549 kB)
----- 0.0/549.3 kB ? eta -:-:-:-
----- 549.3/549.3 kB 3.6 MB/s
eta 0:00:00
Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Downloading numpy-2.0.2-cp312-cp312-win_amd64.whl (15.6 MB)
----- 0.0/15.6 MB ? eta -:-:-:-
----- 1.6/15.6 MB 7.6 MB/s eta
0:00:02
----- 3.1/15.6 MB 7.1 MB/s eta
0:00:02
----- 5.2/15.6 MB 8.4 MB/s eta
0:00:02
----- 7.6/15.6 MB 9.2 MB/s eta
0:00:01
----- 10.2/15.6 MB 9.8 MB/s eta
0:00:01
----- 12.1/15.6 MB 9.6 MB/s eta
0:00:01
----- 13.4/15.6 MB 9.8 MB/s eta
0:00:01
----- 15.5/15.6 MB 9.4 MB/s eta
0:00:01
----- 15.6/15.6 MB 8.8 MB/s eta
0:00:00
Installing collected packages: slicer, numpy, shap
Successfully installed numpy-2.0.2 shap-0.50.0 slicer-0.0.8

WARNING: Retrying (Retry(total=4, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An

```

```
existing connection was forcibly closed by the remote host', None,
10054, None)): /simple/shap/
WARNING: Retrying (Retry(total=3, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An
existing connection was forcibly closed by the remote host', None,
10054, None))': /simple/shap/
WARNING: Retrying (Retry(total=4, connect=None, read=None,
redirect=None, status=None)) after connection broken by
'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An
existing connection was forcibly closed by the remote host', None,
10054, None))':
/packages/c1/29/021db5e35636aae199a40cfdbb87794432bbe9660c5bdbff0867d7
54e400/shap-0.50.0-cp312-cp312-win_amd64.whl.metadata
WARNING: The scripts f2py.exe and numpy-config.exe are installed in
'C:\Users\network-kamc\AppData\Roaming\Python\Python312\Scripts' which
is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress
this warning, use --no-warn-script-location.
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
contourpy 1.2.0 requires numpy<2.0,>=1.20, but you have numpy 2.0.2
which is incompatible.
gensim 4.3.3 requires numpy<2.0,>=1.18.5, but you have numpy 2.0.2
which is incompatible.
```

```
import shap
```

```
pip install shap --no-build-isolation
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: shap in c:\users\network-kamc\appdata\
roaming\python\python312\site-packages (0.50.0)
Requirement already satisfied: numpy>=2 in c:\users\network-kamc\
appdata\roaming\python\python312\site-packages (from shap) (2.0.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\
site-packages (from shap) (1.13.1)
Requirement already satisfied: scikit-learn in c:\programdata\
anaconda3\lib\site-packages (from shap) (1.5.1)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\
site-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in c:\programdata\
anaconda3\lib\site-packages (from shap) (4.66.5)
Requirement already satisfied: packaging>20.9 in c:\programdata\
anaconda3\lib\site-packages (from shap) (24.1)
Requirement already satisfied: slicer==0.0.8 in c:\users\network-kamc\
appdata\roaming\python\python312\site-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in c:\programdata\
```

```

anaconda3\lib\site-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in c:\programdata\
anaconda3\lib\site-packages (from shap) (3.0.0)
Requirement already satisfied: typing-extensions in c:\programdata\
anaconda3\lib\site-packages (from shap) (4.11.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\
programdata\anaconda3\lib\site-packages (from numba>=0.54->shap)
(0.43.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\
lib\site-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\
programdata\anaconda3\lib\site-packages (from pandas->shap)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas->shap) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\
anaconda3\lib\site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.2.0 in c:\programdata\
anaconda3\lib\site-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\programdata\
anaconda3\lib\site-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Note: you may need to restart the kernel to use updated packages.

```

```

from sklearn.inspection import permutation_importance
import pandas as pd
import matplotlib.pyplot as plt

```

```

perm = permutation_importance(
    model,
    X_test,
    y_test,
    n_repeats=10,
    random_state=42,
    scoring="roc_auc"
)

```

```

perm_imp = pd.Series(
    perm.importances_mean,
    index=X_test.columns
).sort_values(ascending=False)

```

```

perm_imp.head(15)

```

```

-----
-----
CatBoostError                                Traceback (most recent call
last)
Cell In[143], line 5

```

```

2 import pandas as pd
3 import matplotlib.pyplot as plt
----> 5 perm = permutation_importance(
6     model,
7     X_test,
8     y_test,
9     n_repeats=10,
10    random_state=42,
11    scoring="roc_auc"
12 )
14 perm_imp = pd.Series(
15     perm.importances_mean,
16     index=X_test.columns
17 ).sort_values(ascending=False)
19 perm_imp.head(15)

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\
_param_validation.py:213, in
validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
207 try:
208     with config_context(
209         skip_parameter_validation=(
210             prefer_skip_nested_validation or
global_skip_validation
211         )
212     ):
--> 213         return func(*args, **kwargs)
214 except InvalidParameterError as e:
215     # When the function is just a wrapper around an estimator,
we allow
216     # the function to delegate validation to the estimator,
but we replace
217     # the name of the estimator by the name of the function in
the error
218     # message to avoid confusion.
219     msg = re.sub(
220         r"parameter of \w+ must be",
221         f"parameter of {func.__qualname__} must be",
222         str(e),
223     )

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\inspection\
_permutation_importance.py:282, in permutation_importance(estimator,
X, y, scoring, n_repeats, n_jobs, random_state, sample_weight,
max_samples)
279     raise ValueError("max_samples must be <= n_samples")
281 scorer = check_scoring(estimator, scoring=scoring)
--> 282 baseline_score = _weights_scorer(scorer, estimator, X, y,
sample_weight)

```

```

284 scores = Parallel(n_jobs=n_jobs)(
285     delayed(_calculate_permutation_scores)(
286         estimator,
287     (...))
288     for col_idx in range(X.shape[1])
289 )
290 if isinstance(baseline_score, dict):

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\inspection_permutation_importance.py:25, in _weights_scorer(scorer, estimator, X, y, sample_weight)

```

23 if sample_weight is not None:
24     return scorer(estimator, X, y,
sample_weight=sample_weight)
--> 25 return scorer(estimator, X, y)

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_scorer.py:279, in _BaseScorer.__call__(self, estimator, X, y_true, sample_weight, **kwargs)

```

276 if sample_weight is not None:
277     _kwargs["sample_weight"] = sample_weight
--> 279 return self._score(partial(_cached_call, None), estimator, X,
y_true, **_kwargs)

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_scorer.py:371, in _Scorer._score(self, method_caller, estimator, X, y_true, **kwargs)

```

369 pos_label = None if is_regressor(estimator) else
self._get_pos_label()
370 response_method = _check_response_method(estimator,
self._response_method)
--> 371 y_pred = method_caller(
372     estimator, response_method.__name__, X,
pos_label=pos_label
373 )
375 scoring_kwargs = {**self._kwargs, **kwargs}
376 return self._sign * self._score_func(y_true, y_pred,
**scoring_kwargs)

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_scorer.py:89, in _cached_call(cache, estimator, response_method, *args, **kwargs)

```

86 if cache is not None and response_method in cache:
87     return cache[response_method]
--> 89 result, _ = _get_response_values(
90     estimator, *args, response_method=response_method,
**kwargs
91 )
93 if cache is not None:
94     cache[response_method] = result

```

```

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\
_response.py:211, in _get_response_values(estimator, X,
response_method, pos_label, return_response_method_used)
    208     elif pos_label is None and target_type == "binary":
    209         pos_label = classes[-1]
-> 211 y_pred = prediction_method(X)
    213 if prediction_method.__name__ in ("predict_proba",
"predict_log_proba"):
    214     y_pred = _process_predict_proba(
    215         y_pred=y_pred,
    216         target_type=target_type,
    217         classes=classes,
    218         pos_label=pos_label,
    219     )

```

```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\
core.py:5351, in CatBoostClassifier.predict_proba(self, X,
ntree_start, ntree_end, thread_count, verbose, task_type)
    5309 def predict_proba(self, X, ntree_start=0, ntree_end=0,
thread_count=-1, verbose=None, task_type="CPU"):
    5310     """
    5311     Predict class probability with X.
    5312     (...)
    5349         with probability for every class for each object.
    5350     """
-> 5351     return self._predict(X, 'Probability', ntree_start,
ntree_end, thread_count, verbose, 'predict_proba', task_type)

```

```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\
core.py:2623, in CatBoost._predict(self, data, prediction_type,
ntree_start, ntree_end, thread_count, verbose, parent_method_name,
task_type)
    2620 data, data_is_single_object =
self._process_predict_input_data(data, parent_method_name,
thread_count)
    2621 self._validate_prediction_type(prediction_type)
-> 2623 predictions = self._base_predict(data, prediction_type,
ntree_start, ntree_end, thread_count, verbose, task_type)
    2624 return predictions[0] if data_is_single_object else
predictions

```

```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\
core.py:1842, in _CatBoostBase._base_predict(self, pool,
prediction_type, ntree_start, ntree_end, thread_count, verbose,
task_type)
    1841 def _base_predict(self, pool, prediction_type, ntree_start,
ntree_end, thread_count, verbose, task_type):
-> 1842     return self._object._base_predict(pool, prediction_type,

```

```
ntree_start, ntree_end, thread_count, verbose, task_type)
File _catboost.pyx:5155, in _catboost._CatBoost._base_predict()
File _catboost.pyx:5162, in _catboost._CatBoost._base_predict()
CatBoostError: catboost/libs/data/model_dataset_compatibility.cpp:81:
At position 22 should be feature with name d1_renal (found
d2_diabetes).
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
cb_importance = pd.Series(
    model.get_feature_importance(),
    index=X_test.columns
).sort_values(ascending=False)
```

```
cb_importance.head(15)
```

```
-----
-----
ValueError                                Traceback (most recent call
last)
```

```
Cell In[145], line 4
```

```
      1 import pandas as pd
      2 import matplotlib.pyplot as plt
----> 4 cb_importance = pd.Series(
      5     model.get_feature_importance(),
      6     index=X_test.columns
      7 ).sort_values(ascending=False)
      9 cb_importance.head(15)
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
series.py:575, in Series.__init__(self, data, index, dtype, name,
copy, fastpath)
```

```
      573     index = default_index(len(data))
      574 elif is_list_like(data):
--> 575     com.require_length_match(data, index)
      577 # create/copy the manager
      578 if isinstance(data, (SingleBlockManager, SingleArrayManager)):
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
common.py:573, in require_length_match(data, index)
```

```
      569 """
      570 Check the length of data matches the length of the index.
      571 """
      572 if len(data) != len(index):
--> 573     raise ValueError(
      574         "Length of values "
```

```

575         f"({len(data)}) "
576         "does not match length of index "
577         f"({len(index)})"
578     )

```

ValueError: Length of values (35) does not match length of index (32)

أسماء الميزات التي يتوقعها نموذج القلب

```

feat_names = model.feature_names_
print(len(feat_names), feat_names[:10])

```

```

35 ['time_in_hospital', 'num_medications', 'num_lab_procedures',
    'num_procedures', 'total_visits', 'major_procedures',
    'number_diagnoses', 'age_[0-10)', 'age_[10-20)', 'age_[20-30)']

```

1) المستخدم للقلب X بناء نفس

```

base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]

```

```

age_dummies_c = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)

```

icd_dummies موجودة (من *cardiac pipeline*) تأكد أن

إذا غير موجودة، أنشئها بنفس الطريقة التي استخدمتها للقلب

```

X_cardiac = pd.concat([df_hosp[base_features], age_dummies_c,
icd_dummies], axis=1)
X_cardiac = X_cardiac.apply(pd.to_numeric,
errors="coerce").fillna(0).astype(int)

```

```

y_cardiac = df_hosp["cardiac_complication"].astype(int)

```

2) المستخدم سابقًا *split* نفس

```

from sklearn.model_selection import train_test_split
Xc_train, Xc_test, yc_train, yc_test = train_test_split(
    X_cardiac, y_cardiac, test_size=0.2, random_state=42,
stratify=y_cardiac
)

```

```

print(Xc_test.shape)

```

```

(20354, 35)

```

```

import pandas as pd

```

```

cb_importance = pd.Series(
    model.get_feature_importance(),
    index=Xc_test.columns
).sort_values(ascending=False)

cb_importance.head(15)

d1_heart_failure      22.052120
d1_ischemic           20.069593
d2_heart_failure      18.852162
d3_heart_failure      13.438002
d3_ischemic           8.986754
d2_ischemic           7.634011
d1_diabetes           0.656956
time_in_hospital      0.625739
age_[40-50)           0.600095
age_[30-40)           0.599494
num_procedures        0.597665
age_[80-90)           0.578842
number_diagnoses      0.573935
num_medications       0.550776
num_lab_procedures    0.480760
dtype: float64

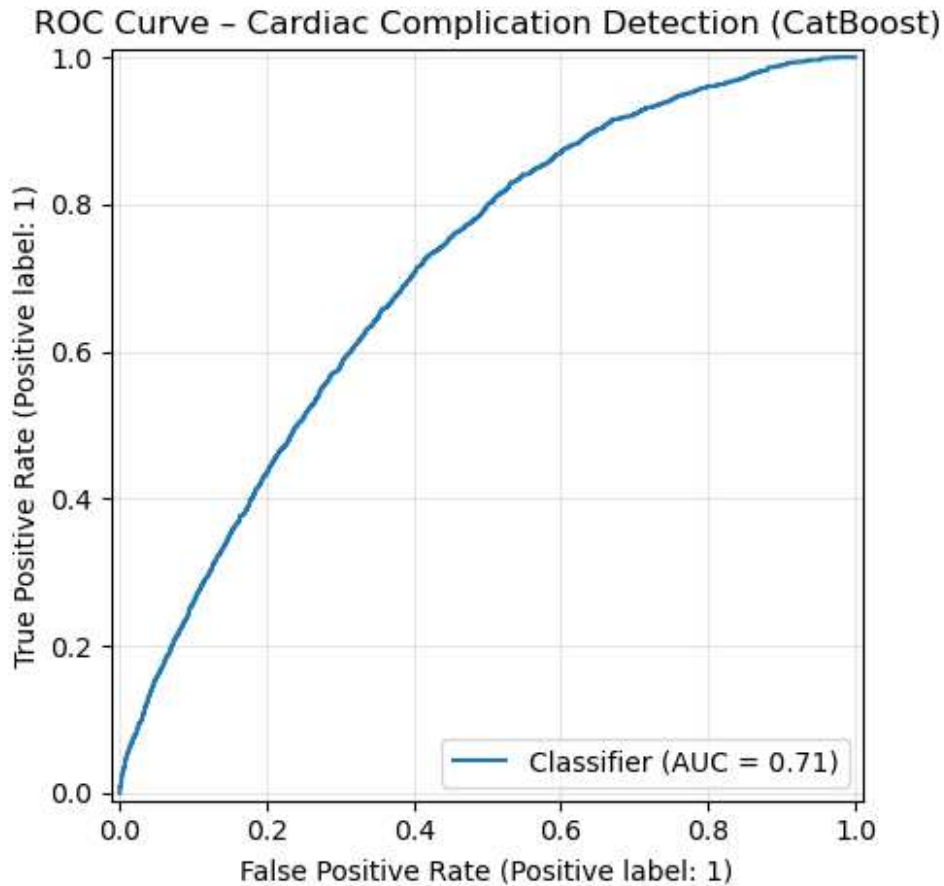
print("Model features:", len(model.get_feature_importance()))
print("X_test columns:", X_test.shape[1])

Model features: 35
X_test columns: 32

from sklearn.metrics import RocCurveDisplay
import matplotlib.pyplot as plt

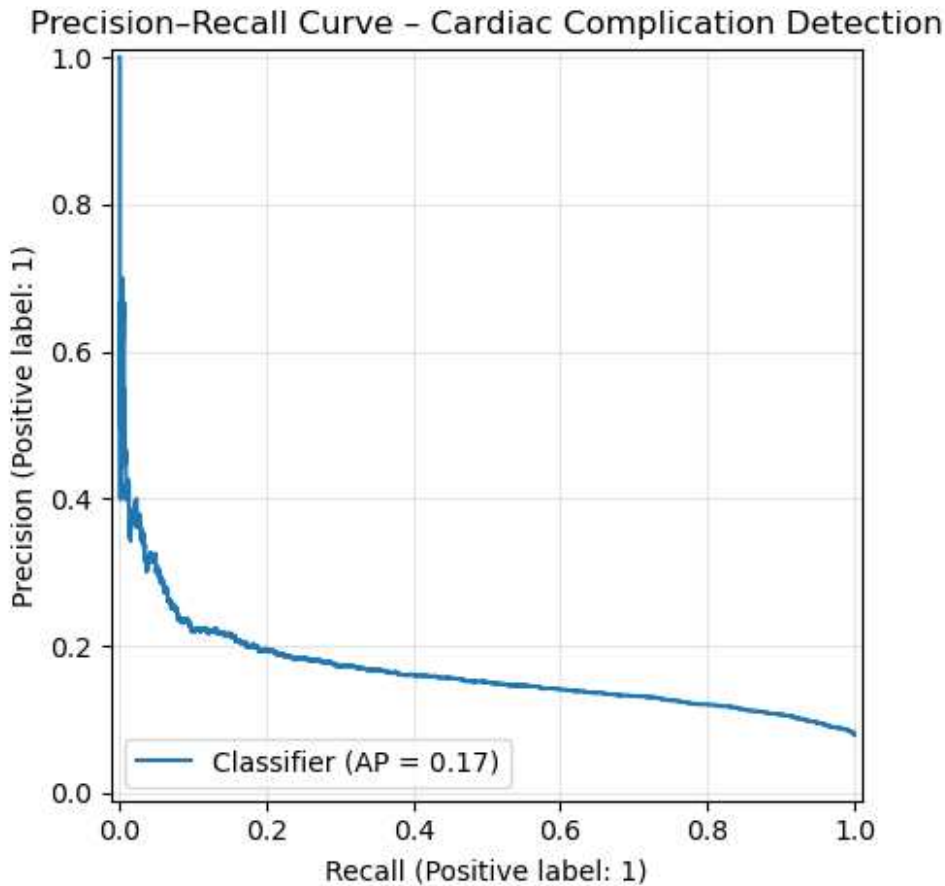
RocCurveDisplay.from_predictions(y_test, y_prob)
plt.title("ROC Curve – Cardiac Complication Detection (CatBoost)")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()

```



```
from sklearn.metrics import PrecisionRecallDisplay
import matplotlib.pyplot as plt

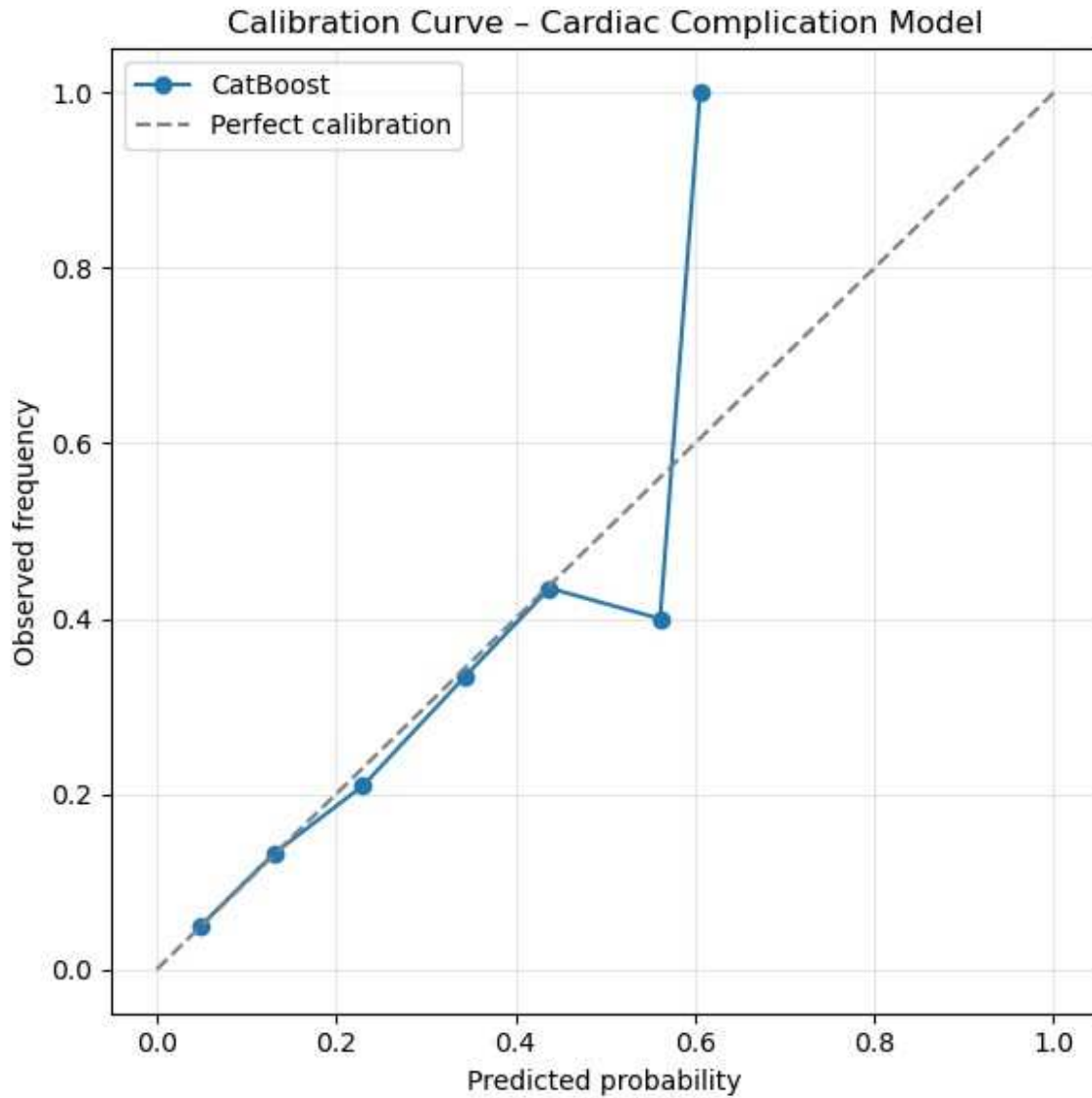
PrecisionRecallDisplay.from_predictions(y_test, y_prob)
plt.title("Precision-Recall Curve – Cardiac Complication Detection")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



```
from sklearn.calibration import calibration_curve
import matplotlib.pyplot as plt

prob_true, prob_pred = calibration_curve(y_test, y_prob, n_bins=10,
strategy="uniform")

plt.figure(figsize=(6,6))
plt.plot(prob_pred, prob_true, marker="o", label="CatBoost")
plt.plot([0,1], [0,1], linestyle="--", color="gray", label="Perfect
calibration")
plt.xlabel("Predicted probability")
plt.ylabel("Observed frequency")
plt.title("Calibration Curve – Cardiac Complication Model")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



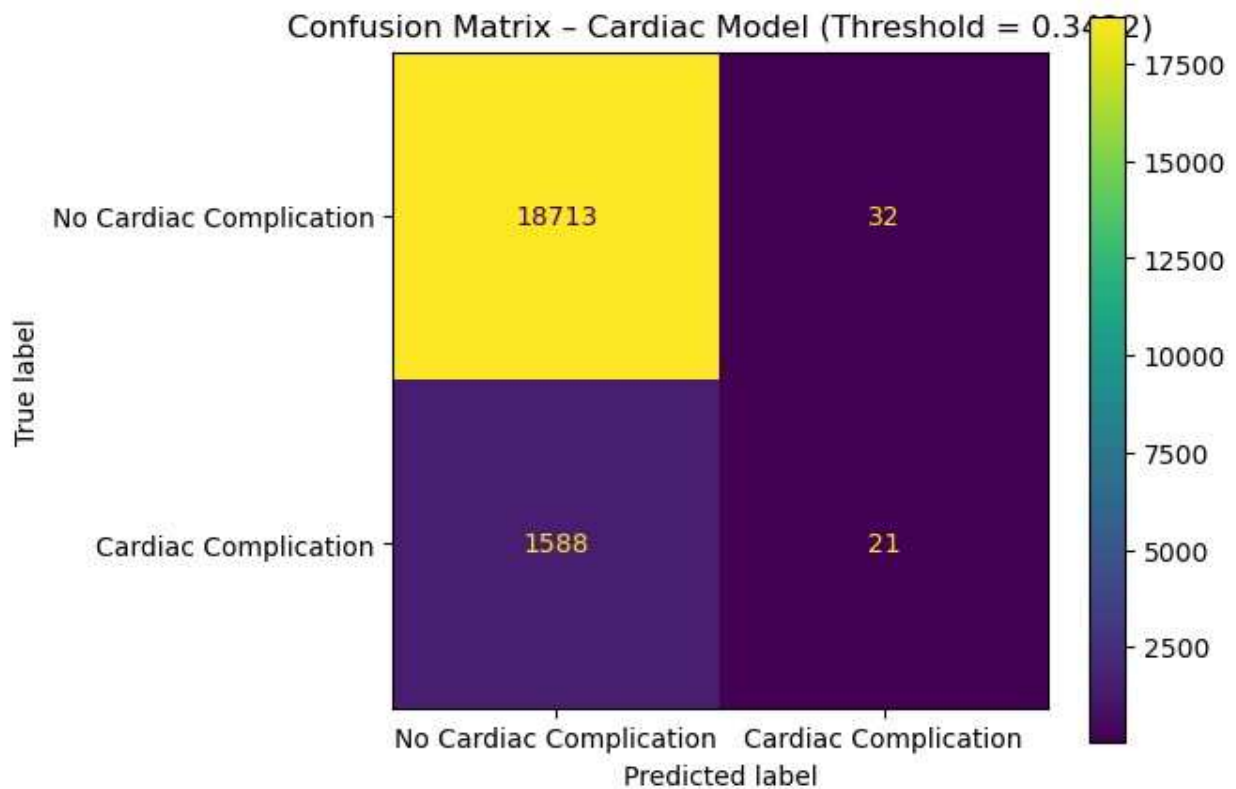
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

best_thr = 0.3422
y_pred = (y_prob >= best_thr).astype(int)

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=["No Cardiac Complication", "Cardiac Complication"]
)
disp.plot(values_format="d")
plt.title("Confusion Matrix - Cardiac Model (Threshold = 0.3422)")
```

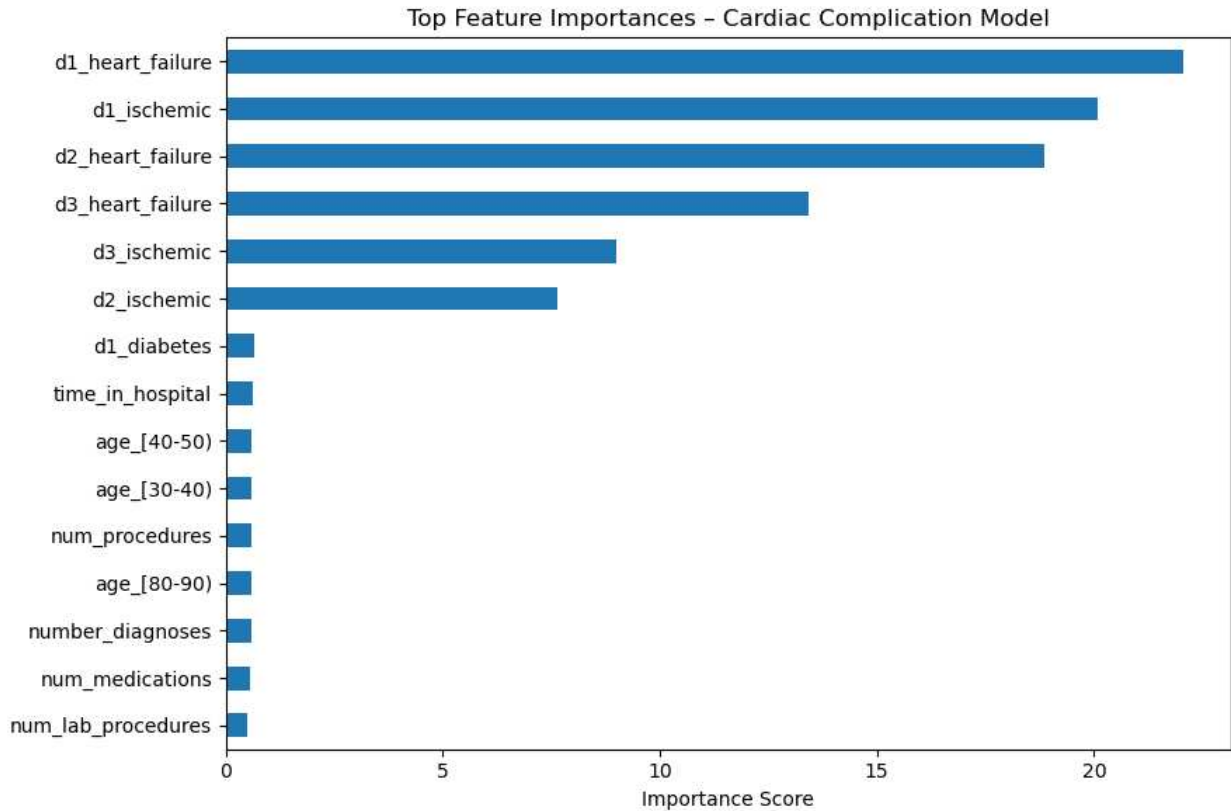
```
plt.tight_layout()
plt.show()
```



```
import matplotlib.pyplot as plt

top15 = cb_importance.head(15)

plt.figure(figsize=(9,6))
top15.sort_values().plot(kind="barh")
plt.title("Top Feature Importances – Cardiac Complication Model")
plt.xlabel("Importance Score")
plt.tight_layout()
plt.show()
```



```
print("len(y_prob) =", len(y_prob))
print("X_test shape =", X_test.shape)
print("y_test shape =", y_test.shape)
```

```
len(y_prob) = 20354
X_test shape = (20354, 32)
y_test shape = (20354,)
```

```
import shap
```

```
# مهم: TreeExplainer مع CatBoost
explainer = shap.TreeExplainer(model)
```

```
# حساب SHAP values
shap_values = explainer.shap_values(X_test)
```

```
-----
-----
CatBoostError Traceback (most recent call
last)
```

```
Cell In[167], line 7
```

```
4 explainer = shap.TreeExplainer(model)
```

```
6 # حساب SHAP values
```

```
----> 7 shap_values = explainer.shap_values(X_test)
```

```

File ~\AppData\Roaming\Python\Python312\site-packages\shap\explainers\
_tree.py:607, in TreeExplainer.shap_values(self, X, y, tree_limit,
approximate, check_additivity, from_call)
    605     if not isinstance(X, catboost.Pool):
    606         X = catboost.Pool(X,
cat_features=self.model.cat_feature_indices)
--> 607     phi =
self.model.original_model.get_feature_importance(data=X,
fstr_type="ShapValues")
    609 # note we pull off the last column and keep it as our
expected_value
    610 if phi is not None:

```

```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\
core.py:3249, in CatBoost.get_feature_importance(self, data, type,
prettified, thread_count, verbose, fstr_type, shap_mode, model_output,
interaction_indices, shap_calc_type, reference_data, sage_n_samples,
sage_batch_size, sage_detect_convergence, log_cout, log_cerr)
    3246     raise CatBoostError("data is empty.")
    3248 shap_calc_type = enum_from_enum_or_str(EShapCalcType,
shap_calc_type).value
-> 3249 fstr, feature_names = self._calc_fstr(type, data,
reference_data, thread_count, verbose, model_output, shap_mode,
interaction_indices,
    3250                                     shap_calc_type,
sage_n_samples, sage_batch_size, sage_detect_convergence)
    3251 if type in (EFstrType.PredictionValuesChange,
EFstrType.LossFunctionChange,
    3252             EFstrType.PredictionDiff, EFstrType.SageValues):
    3253     feature_importances = [value[0] for value in fstr]

```

```

File ~\AppData\Roaming\Python\Python312\site-packages\catboost\
core.py:1875, in _CatBoostBase._calc_fstr(self, type, pool,
reference_data, thread_count, verbose, model_output, shap_mode,
interaction_indices, shap_calc_type, sage_n_samples, sage_batch_size,
sage_detect_convergence)
    1860 def _calc_fstr(
    1861     self,
    1862     type,
    (...)
    1873     sage_detect_convergence
    1874 ):
-> 1875     return self._object._calc_fstr(
    1876         type.name,
    1877         pool,
    1878         reference_data,
    1879         thread_count,
    1880         verbose,
    1881         model_output,

```

```

1882         shap_mode,
1883         interaction_indices,
1884         shap_calc_type,
1885         sage_n_samples,
1886         sage_batch_size,
1887         sage_detect_convergence
1888     )

```

File _catboost.pyx:5257, in _catboost._CatBoost._calc_fstr()

File _catboost.pyx:5322, in _catboost._CatBoost._calc_fstr()

CatBoostError: catboost/libs/data/model_dataset_compatibility.cpp:81:
At position 22 should be feature with name d1_renal (found
d2_diabetes).

```
import pandas as pd
```

```
# base features
```

```
base_features = [
    "time_in_hospital",
    "num_medications",
    "num_lab_procedures",
    "num_procedures",
    "total_visits",
    "major_procedures",
    "number_diagnoses"
]
```

```
# age dummies (كما في القلب)
```

```
age_dummies_c = pd.get_dummies(df_hosp["age"], prefix="age",
drop_first=False)
```

```
# icd_dummies: لازم تكون هي نفسها التي استخدمتها في القلب (d1/d2/d3 groups)
# إذا كانت موجودة عندك من تجربة القلب، استخدمها مباشرة:
```

```
X_cardiac = pd.concat([df_hosp[base_features], age_dummies_c,
icd_dummies], axis=1)
X_cardiac = X_cardiac.apply(pd.to_numeric,
errors="coerce").fillna(0).astype(int)
```

```
y_cardiac = df_hosp["cardiac_c"]()
```

```
Cell In[169], line 22
```

```
y_cardiac = df_hosp["cardiac_c"]()
```

SyntaxError: unterminated string literal (detected at line 22)

```
from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score
```

```

model_cardiac = CatBoostClassifier(
    iterations=3000,
    learning_rate=0.03,
    depth=7,
    l2_leaf_reg=8,
    loss_function="Logloss",
    eval_metric="AUC",
    random_seed=42,
    od_type="Iter",
    od_wait=200,
    verbose=200
)

model_cardiac.fit(
    Xc_train, yc_train,
    eval_set=(Xc_test, yc_test),
    use_best_model=True
)

y_prob_c = model_cardiac.predict_proba(Xc_test)[: , 1]
print("Cardiac AUC (recheck) =", round(roc_auc_score(yc_test,
y_prob_c), 4))

0:   test: 0.8903095 best: 0.8903095 (0)   total: 27.8ms   remaining:
1m 23s
200: test: 0.9365952 best: 0.9365952 (200) total: 2.46s   remaining:
34.2s
400: test: 0.9377176 best: 0.9377192 (399) total: 4.77s   remaining:
30.9s
600: test: 0.9379685 best: 0.9379766 (570) total: 6.95s   remaining:
27.8s
Stopped by overfitting detector (200 iterations wait)

bestTest = 0.9379766265
bestIteration = 570

Shrink model to first 571 iterations.
Cardiac AUC (recheck) = 0.938

X_shap = Xc_test.sample(2000, random_state=42)

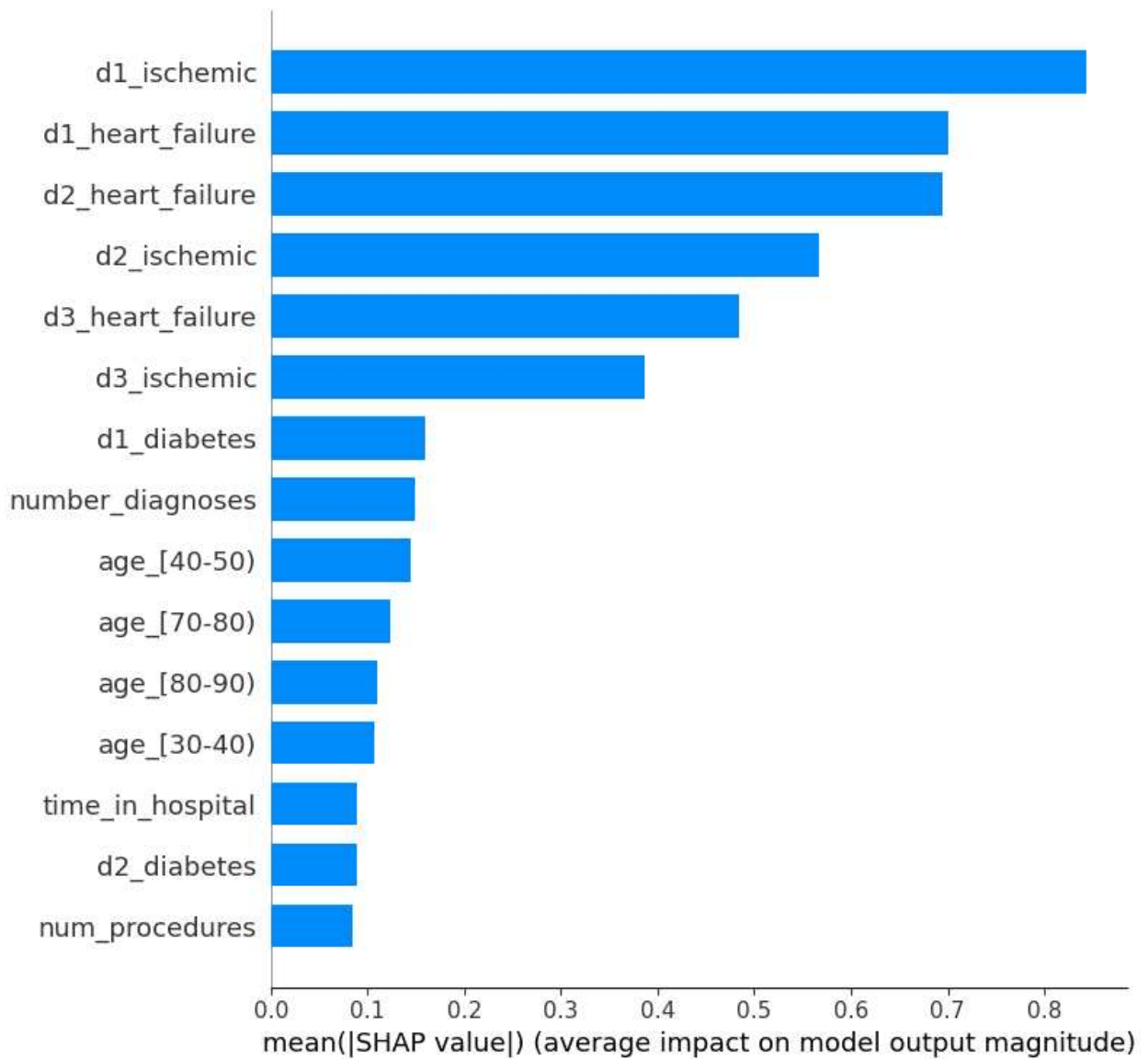
import shap

explainer = shap.TreeExplainer(model_cardiac)
shap_values = explainer.shap_values(X_shap)

shap.summary_plot(
    shap_values,
    X_shap,
    plot_type="bar",

```

```
)  
    max_display=15  
)
```



```
shap.summary_plot(  
    shap_values,  
    X_shap,  
    max_display=15  
)
```

