

```

In [7]: import os, glob
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    classification_report, confusion_matrix, ConfusionMatrixDisplay,
    roc_auc_score, roc_curve, precision_recall_curve, average_precision_score
)
from sklearn.feature_selection import SelectKBest, chi2, RFE

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

# =====
# 1) Robust CSV Loader (prefers heart.csv in current folder)
# =====
def find_csv():
    # Prefer "heart.csv" in current folder; else scan current folder for .csv; else
    local_heart = [p for p in glob.glob("*.csv")+glob.glob("*.CSV") if "heart" in p]
    if local_heart:
        return sorted(local_heart, key=len)[0]
    local_any = glob.glob("*.csv")+glob.glob("*.CSV")
    if local_any:
        return local_any[0]
    downloads = os.path.join(os.path.expanduser("~"), "Downloads")
    rec = glob.glob(os.path.join(downloads, "**", "*.csv"), recursive=True)
    rec_heart = [p for p in rec if "heart" in os.path.basename(p).lower()]
    if rec_heart:
        return sorted(rec_heart, key=len)[0]
    if rec:
        return rec[0]
    raise FileNotFoundError("No CSV found. Put 'heart.csv' next to the notebook or

csv_path = find_csv()
print("Using file:", csv_path)
df = pd.read_csv(csv_path)
print("Data shape:", df.shape)
display(df.head())

# =====
# 2) Target & Basic Prep (UCI 'num' -> binary)
# =====

```

```

target_col = "num"
if target_col not in df.columns:
    raise ValueError(f"Target '{target_col}' not found. Columns: {list(df.columns)}")

X = df.drop(columns=[target_col])
y = (df[target_col] > 0).astype(int)  # 0..4 -> binary: disease > 0

# =====
# 3) Preprocessing (Missing values + encoding; MinMax for chi2 non-negativity)
# =====
num_cols = X.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = [c for c in X.columns if c not in num_cols]

num_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", MinMaxScaler())  # keep numeric >= 0 for chi2
])

cat_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore", sparse_output=False))  # dens
])

preprocessor = ColumnTransformer([
    ("num", num_pipe, num_cols),
    ("cat", cat_pipe, cat_cols)
])

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

# =====
# 5) Models + GridSearch (Optimization)
# =====
models = {
    "LogisticRegression": (
        LogisticRegression(max_iter=5000),
        {"clf__C": [0.1, 1, 10], "clf__solver": ["lbfgs"]}
    ),
    "RandomForest": (
        RandomForestClassifier(random_state=42),
        {"clf__n_estimators": [150, 300], "clf__max_depth": [None, 8, 16]}
    ),
    "NaiveBayes": (
        GaussianNB(),
        {"clf__var_smoothing": [1e-9, 1e-8, 1e-7]}
    ),
    "KNN": (
        KNeighborsClassifier(),
        {"clf__n_neighbors": [3, 5, 11], "clf__weights": ["uniform", "distance"]}
    ),
}

```

```

    "SVM": (
        SVC(probability=True, random_state=42),
        {"clf__C": [0.5, 1, 2], "clf__kernel": ["linear", "rbf"], "clf__gamma": ["s
    )
}

cv = StratifiedKFold(5, shuffle=True, random_state=42)
summary_rows = [] # (phase, model, cv_f1)

def eval_and_plot(name, estimator):
    y_pred = estimator.predict(X_test)
    print(f"\n{name}")
    print(classification_report(y_test, y_pred, digits=4))

    cm = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(cm).plot(values_format='d')
    plt.title(name); plt.show()

    # Curves if proba available
    try:
        y_proba = estimator.predict_proba(X_test)[: , 1]
        roc = roc_auc_score(y_test, y_proba)
        fpr, tpr, _ = roc_curve(y_test, y_proba)
        plt.plot(fpr, tpr, label=f"AUC={roc:.3f}")
        plt.plot([0,1],[0,1], '--')
        plt.xlabel("FPR"); plt.ylabel("TPR")
        plt.title(f"ROC - {name}"); plt.legend(); plt.show()

        prec, rec, _ = precision_recall_curve(y_test, y_proba)
        ap = average_precision_score(y_test, y_proba)
        plt.plot(rec, prec, label=f"AP={ap:.3f}")
        plt.xlabel("Recall"); plt.ylabel("Precision")
        plt.title(f"PR - {name}"); plt.legend(); plt.show()
    except Exception:
        pass

# =====
# 6) FULL FEATURES + GridSearch (First stage REQUIRED)
# =====
print("\n=== FULL FEATURES (no FS) + GridSearch ===")
for name, (model, grid) in models.items():
    pipe = Pipeline([("pre", preprocessor), ("clf", model)])
    gs = GridSearchCV(pipe, grid, cv=cv, scoring="f1", n_jobs=-1, refit=True)
    gs.fit(X_train, y_train)
    eval_and_plot(f"Baseline - {name}", gs.best_estimator_)
    summary_rows.append(("baseline", name, gs.best_score_))

# =====
# 7) FEATURE SELECTION #1 - Filter: Chi-Square (SelectKBest)
# (matches Lecture SelectKBest(chi2)) ← see slides
# =====
k = min(8, max(3, X_train.shape[1] // 2)) # you can change k if your prof specifies
fs_chi2 = SelectKBest(score_func=chi2, k=k)

```

```

print("\n=== FEATURE SELECTION: SelectKBest(chi2) ===")
for name, (model, grid) in models.items():
    pipe = Pipeline([("pre", preprocessor), ("select", fs_chi2), ("clf", model)])
    gs = GridSearchCV(pipe, grid, cv=cv, scoring="f1", n_jobs=-1, refit=True)
    gs.fit(X_train, y_train)
    eval_and_plot(f"Chi2 - {name}", gs.best_estimator_)
    summary_rows.append(("fs_chi2", name, gs.best_score_))

# =====
# 8) FEATURE SELECTION #2 - Wrapper: RFE
# (matches lecture RFE with a model) ← see slides
# =====
# We'll use RF as the RFE estimator (robust on mixed/one-hot features).
rfe_base_estimator = RandomForestClassifier(random_state=42)
rfe = RFE(estimator=rfe_base_estimator, n_features_to_select=k, step=1)

print("\n=== FEATURE SELECTION: RFE (with RandomForest) ===")
for name, (model, grid) in models.items():
    pipe = Pipeline([("pre", preprocessor), ("rfe", rfe), ("clf", model)])
    gs = GridSearchCV(pipe, grid, cv=cv, scoring="f1", n_jobs=-1, refit=True)
    gs.fit(X_train, y_train)
    eval_and_plot(f"RFE - {name}", gs.best_estimator_)
    summary_rows.append(("fs_rfe", name, gs.best_score_))

# =====
# 9) Save Summary
# =====
summary = pd.DataFrame(summary_rows, columns=["phase", "model", "cv_f1"])
summary.sort_values(["phase", "cv_f1"], ascending=[True, False], inplace=True)
summary.to_csv("heart_results_summary.csv", index=False)
print("\nSaved summary to heart_results_summary.csv")
display(summary)

```

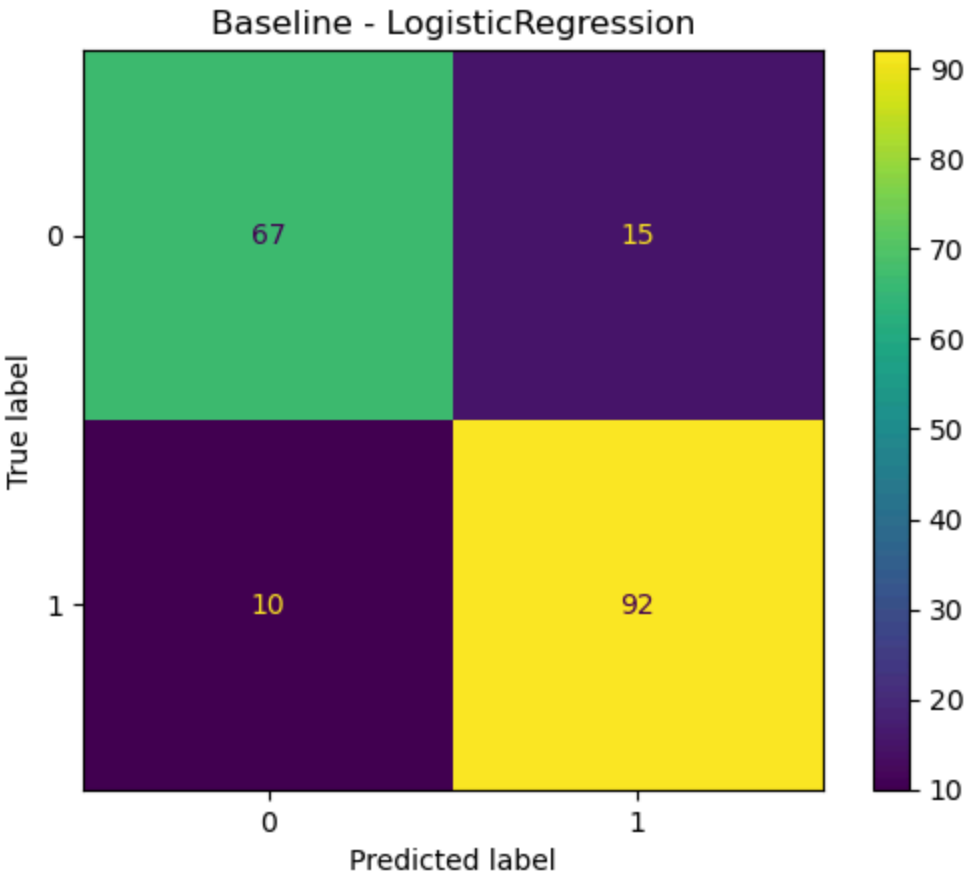
Using file: heart.csv.csv

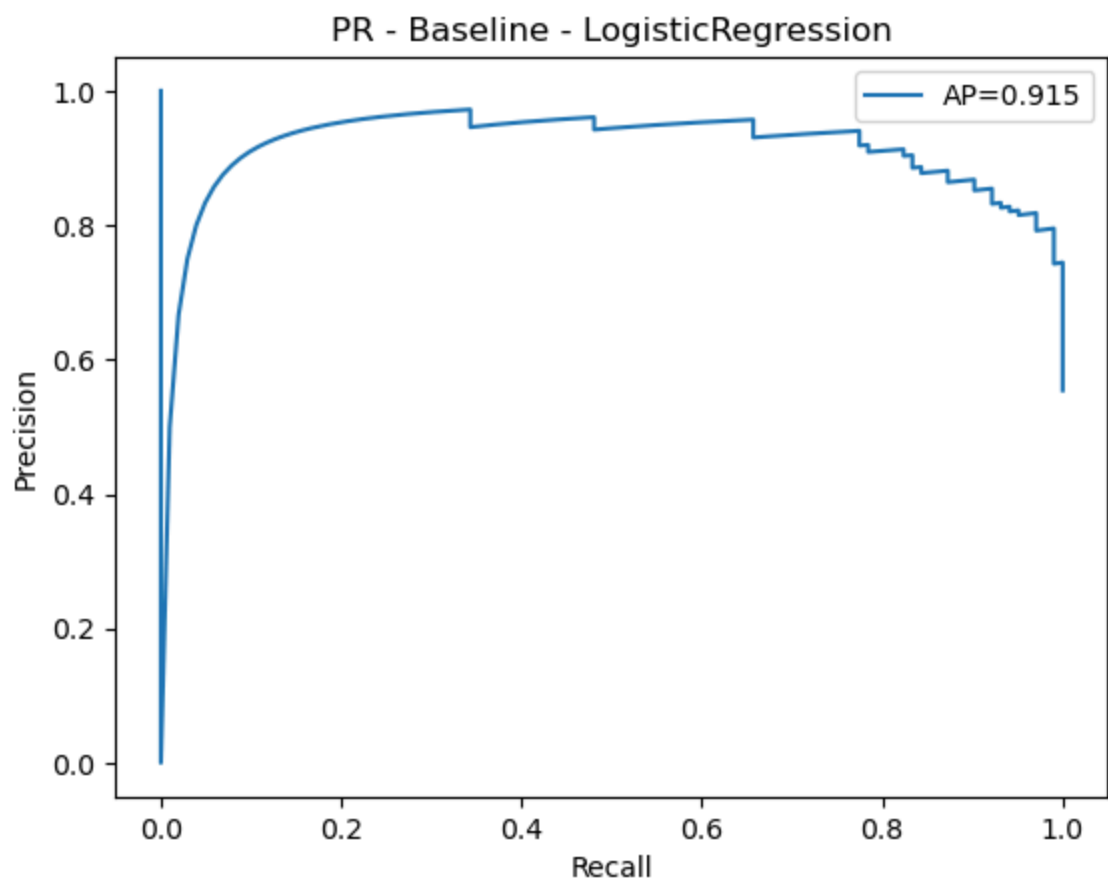
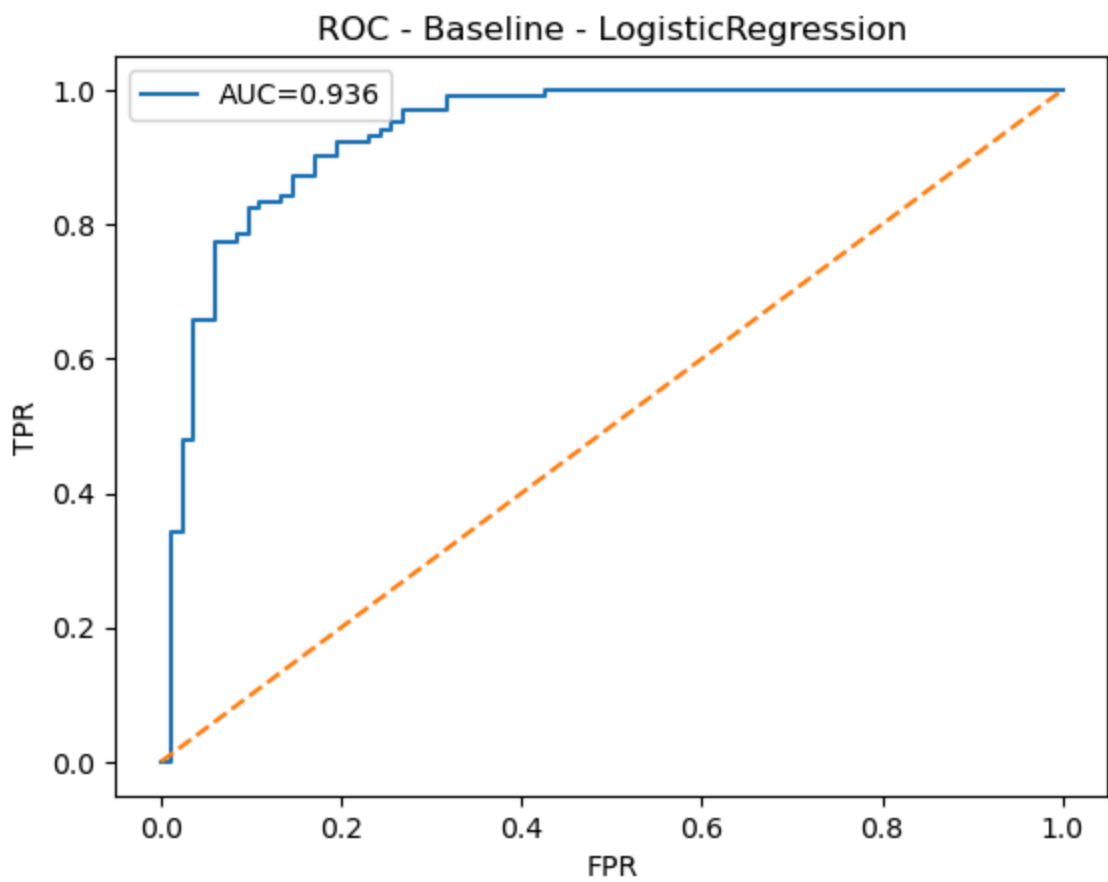
Data shape: (920, 16)

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exar
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	Fal
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	Tr
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	Tr
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	Fal
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	Fal

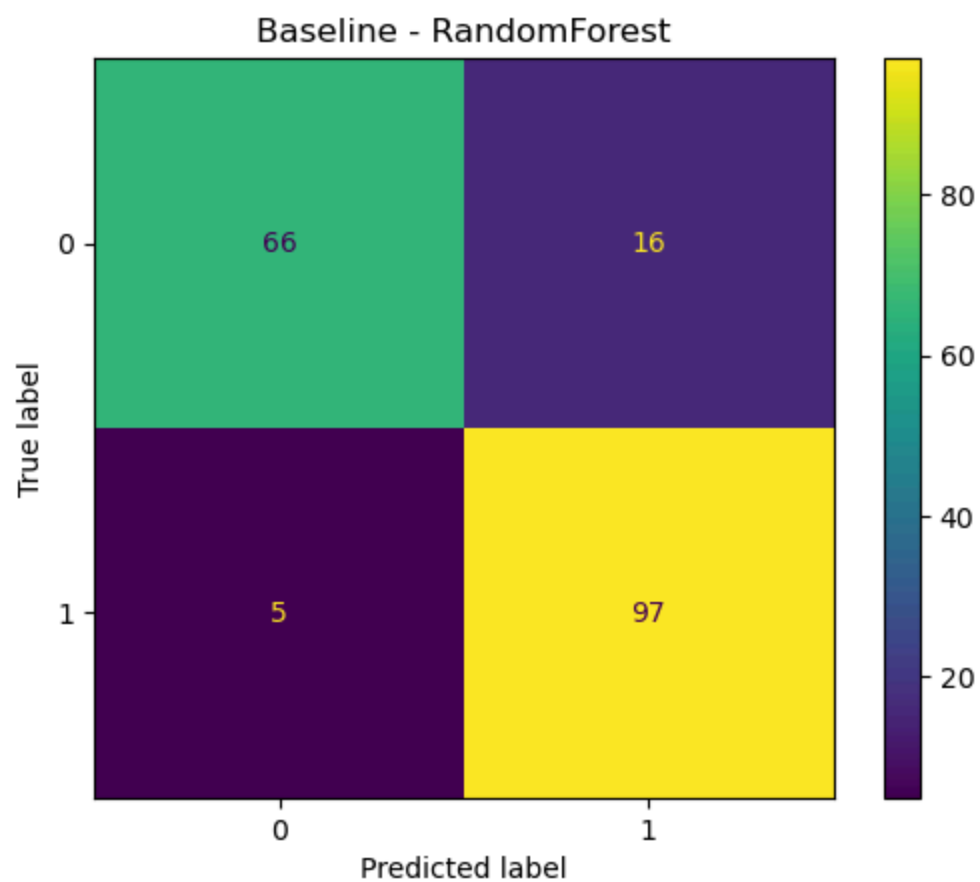
=== FULL FEATURES (no FS) + GridSearch ===

Baseline - LogisticRegression					
	precision	recall	f1-score	support	
0	0.8701	0.8171	0.8428	82	
1	0.8598	0.9020	0.8804	102	
accuracy			0.8641	184	
macro avg	0.8650	0.8595	0.8616	184	
weighted avg	0.8644	0.8641	0.8636	184	

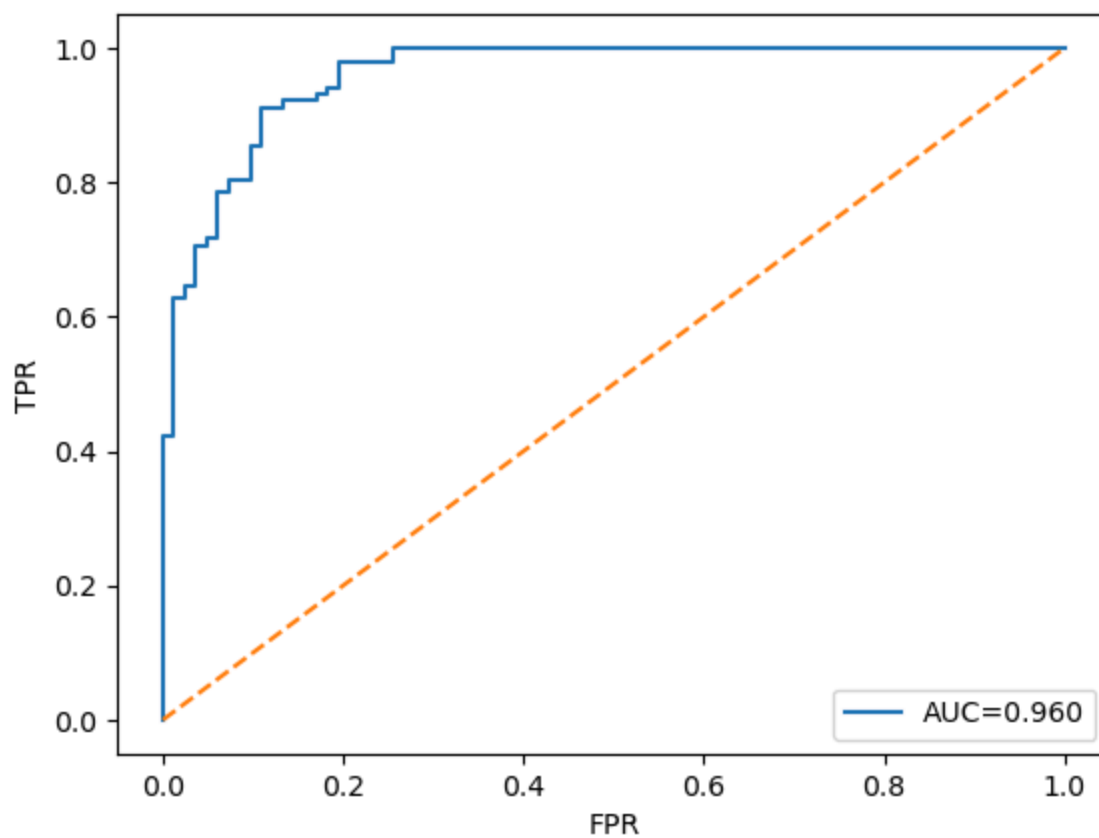




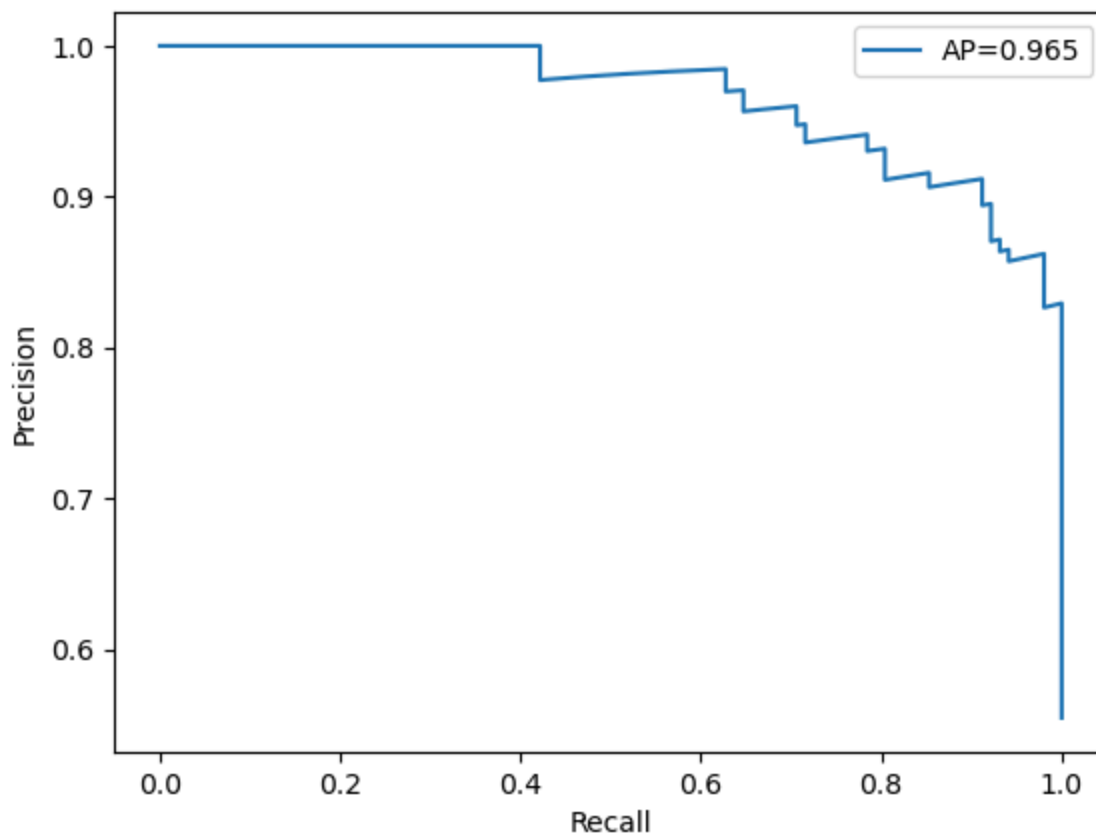
Baseline - RandomForest					
	precision	recall	f1-score	support	
0	0.9296	0.8049	0.8627	82	
1	0.8584	0.9510	0.9023	102	
accuracy			0.8859	184	
macro avg	0.8940	0.8779	0.8825	184	
weighted avg	0.8901	0.8859	0.8847	184	



ROC - Baseline - RandomForest

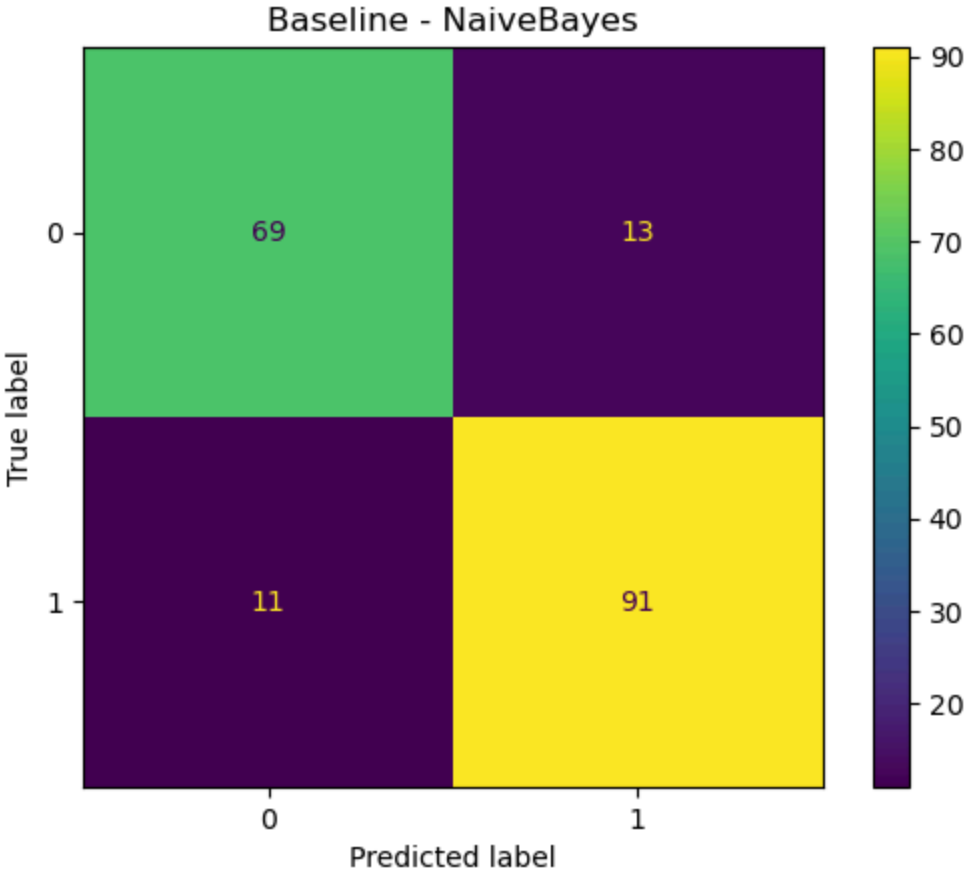


PR - Baseline - RandomForest

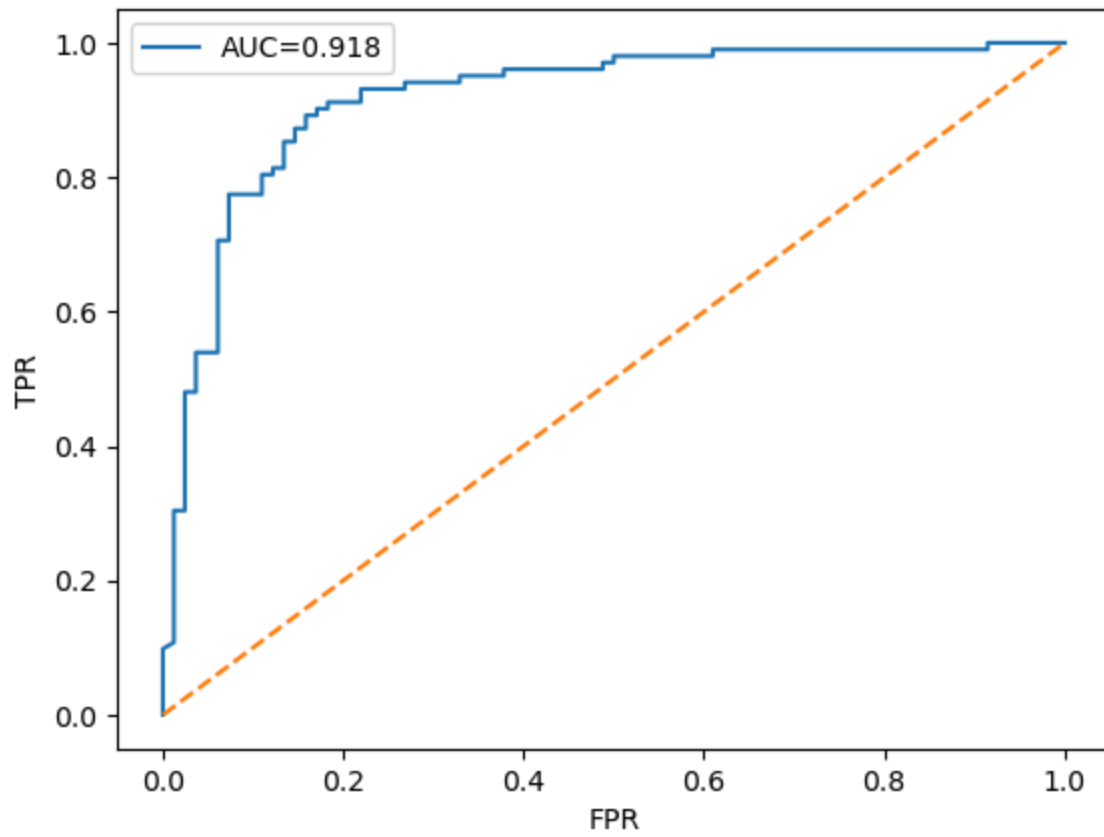


Baseline - NaiveBayes

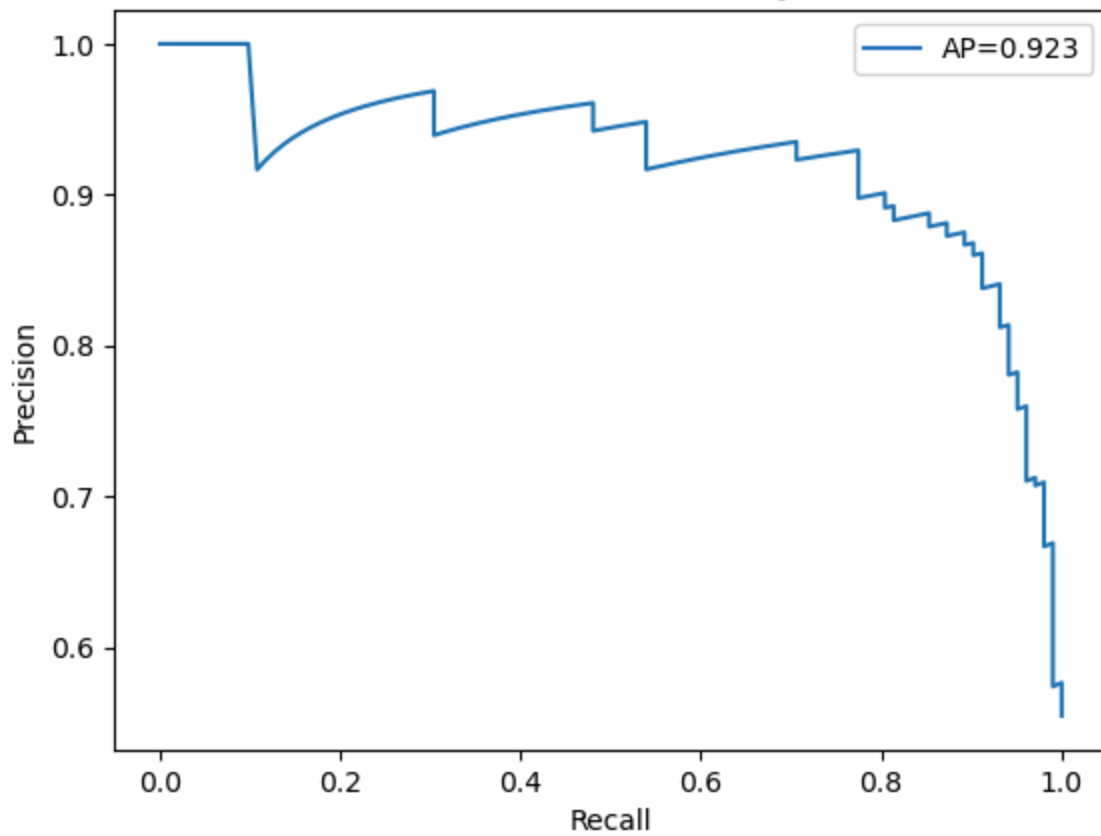
	precision	recall	f1-score	support
0	0.8625	0.8415	0.8519	82
1	0.8750	0.8922	0.8835	102
accuracy			0.8696	184
macro avg	0.8688	0.8668	0.8677	184
weighted avg	0.8694	0.8696	0.8694	184



ROC - Baseline - NaiveBayes

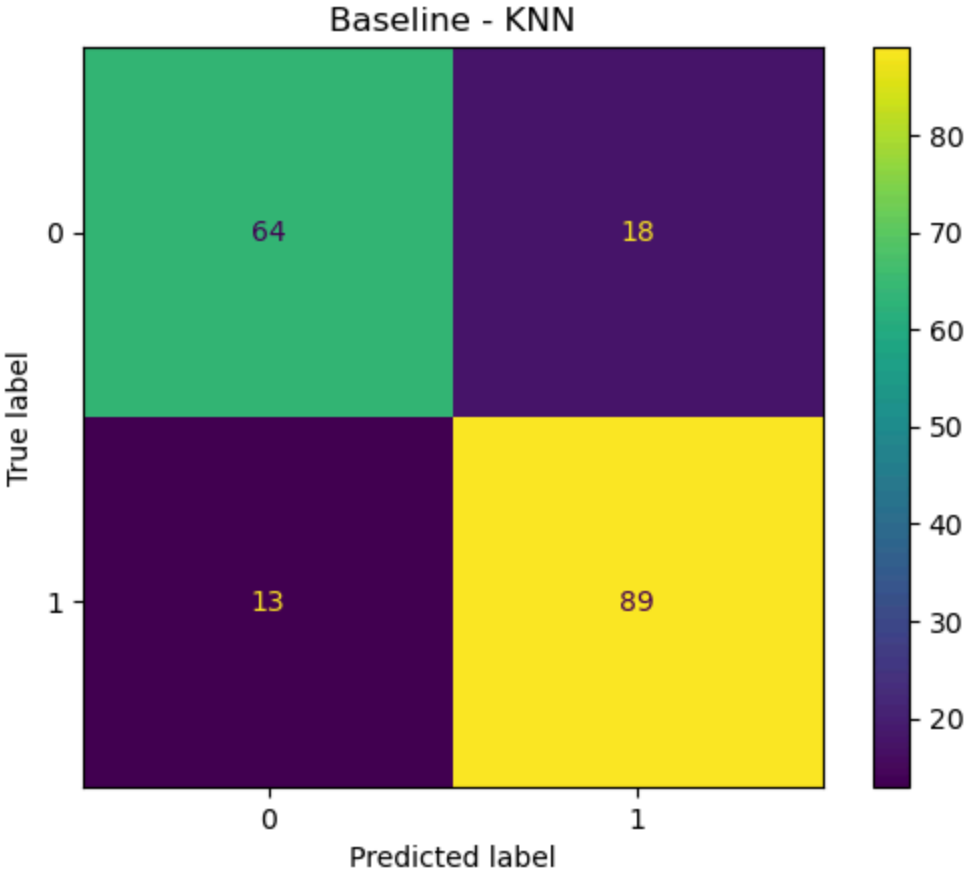


PR - Baseline - NaiveBayes

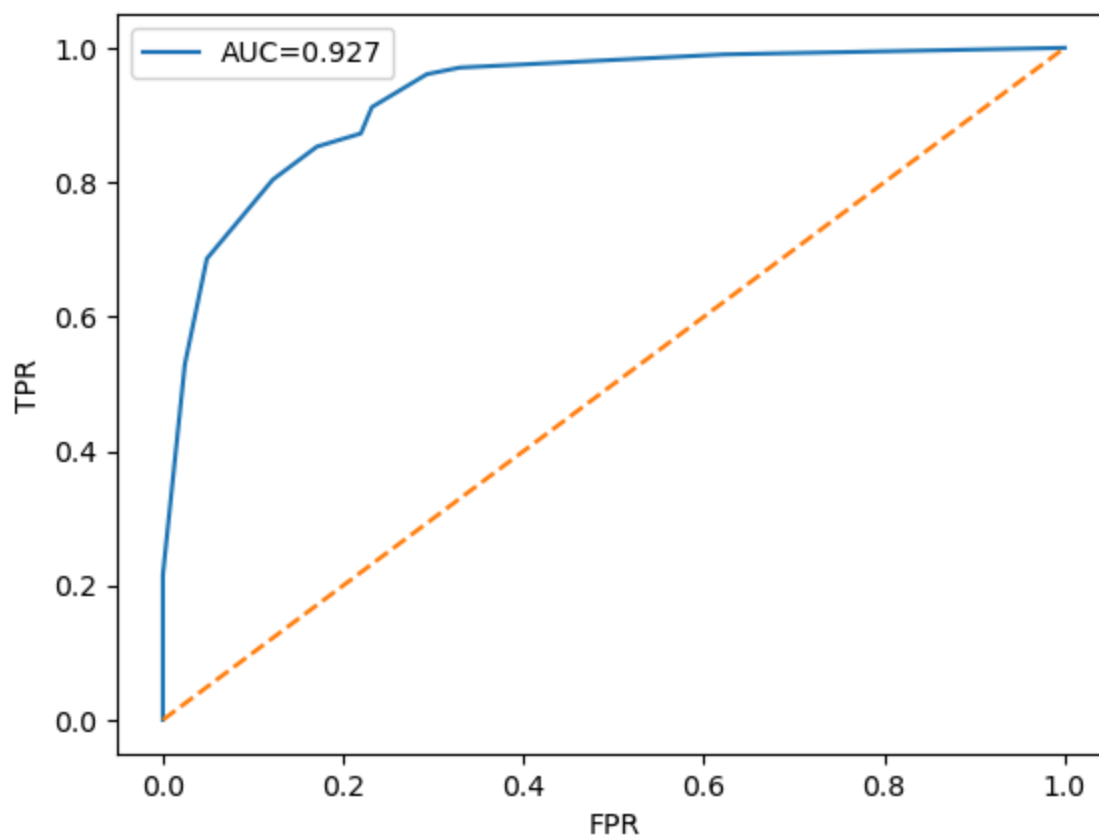


Baseline - KNN

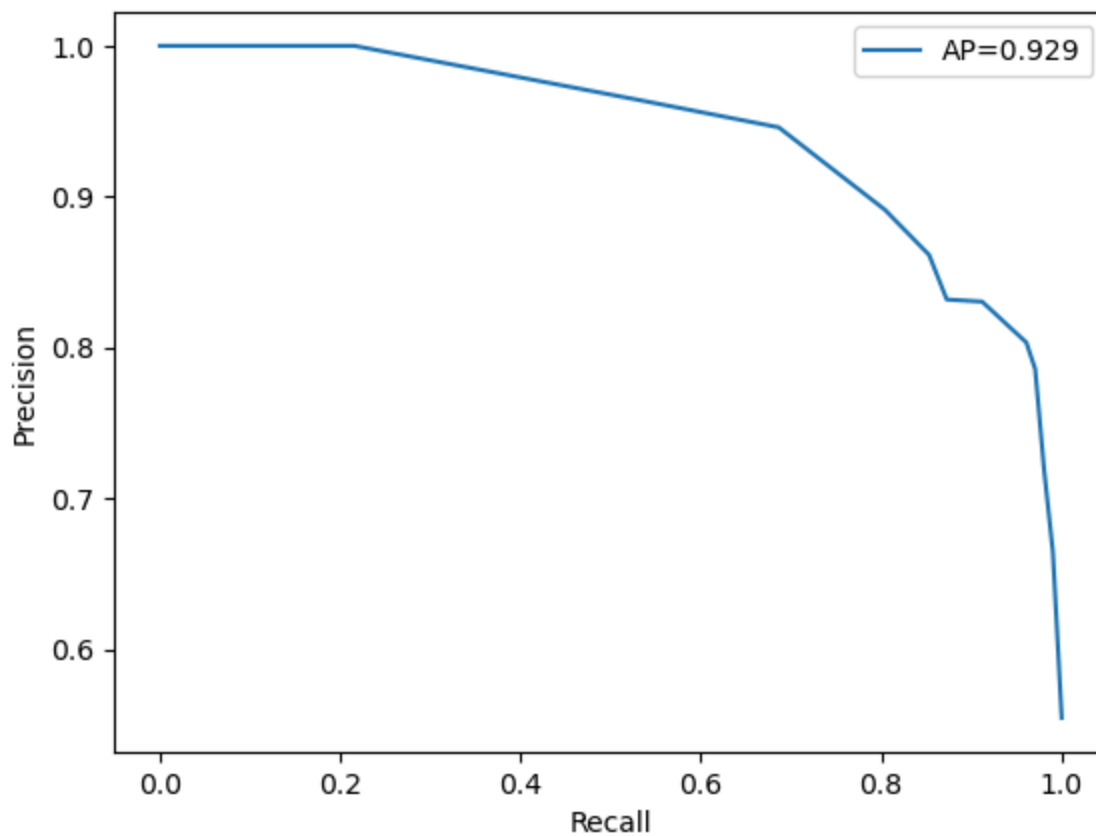
	precision	recall	f1-score	support
0	0.8312	0.7805	0.8050	82
1	0.8318	0.8725	0.8517	102
accuracy			0.8315	184
macro avg	0.8315	0.8265	0.8284	184
weighted avg	0.8315	0.8315	0.8309	184



ROC - Baseline - KNN

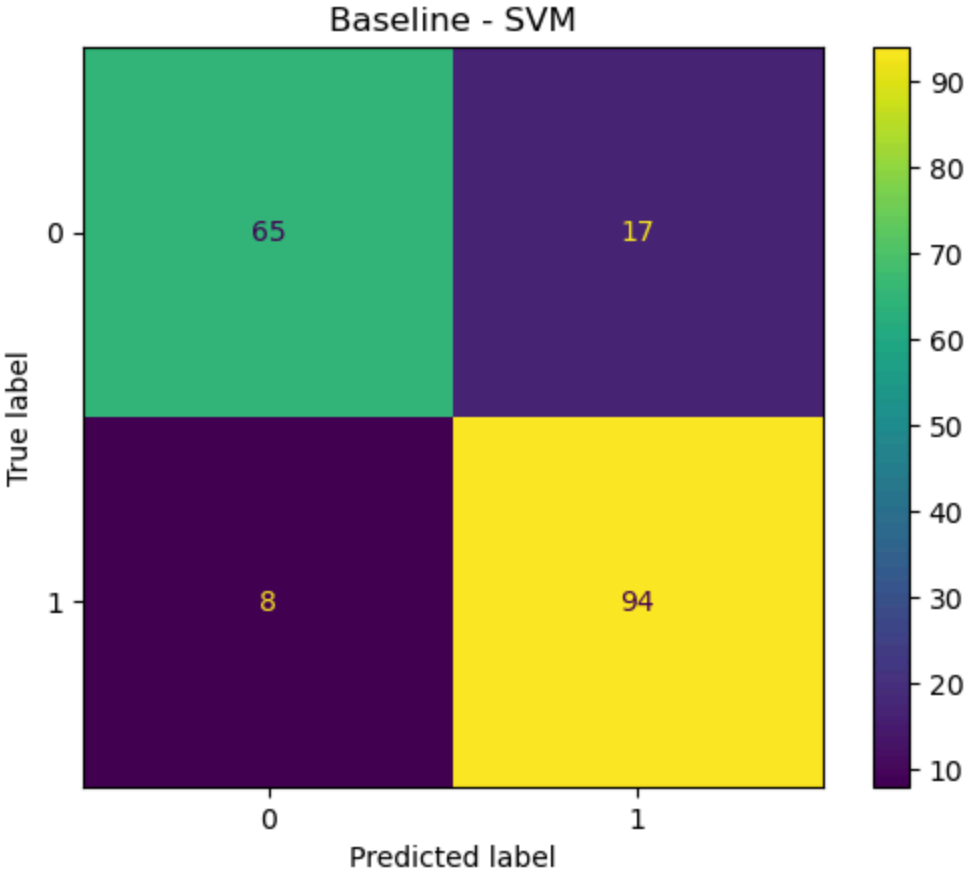


PR - Baseline - KNN

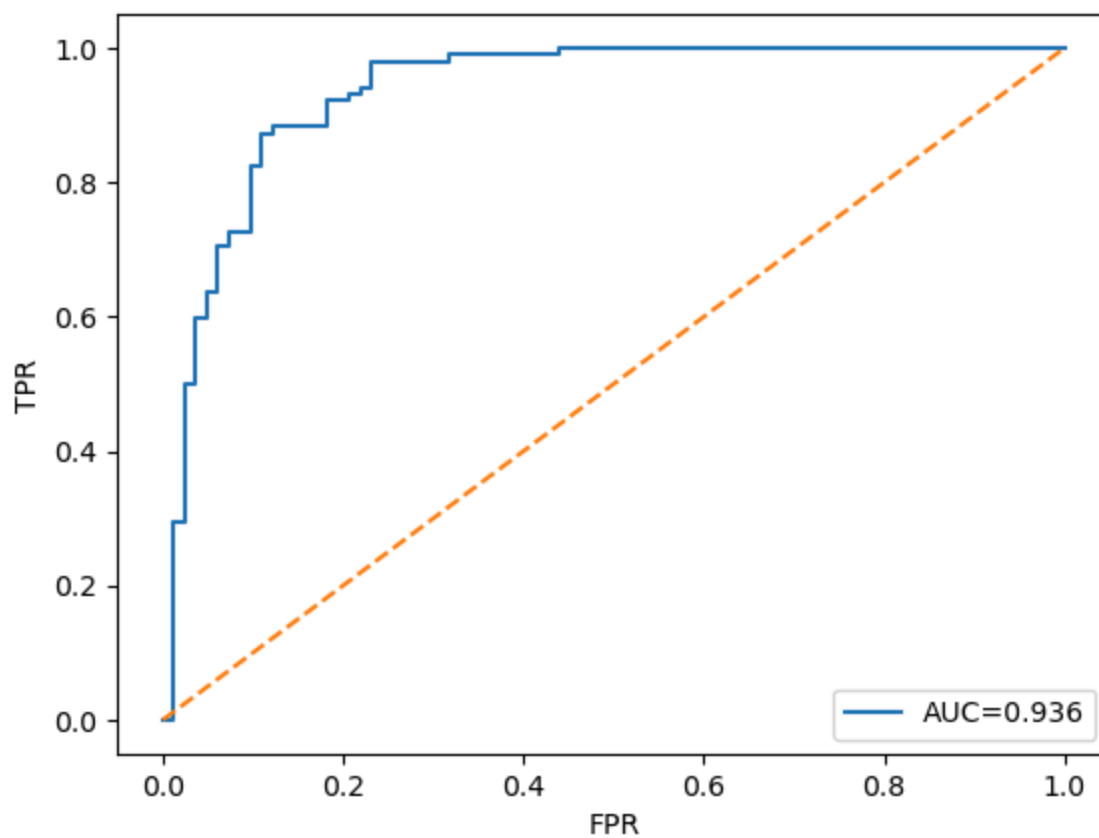


Baseline - SVM

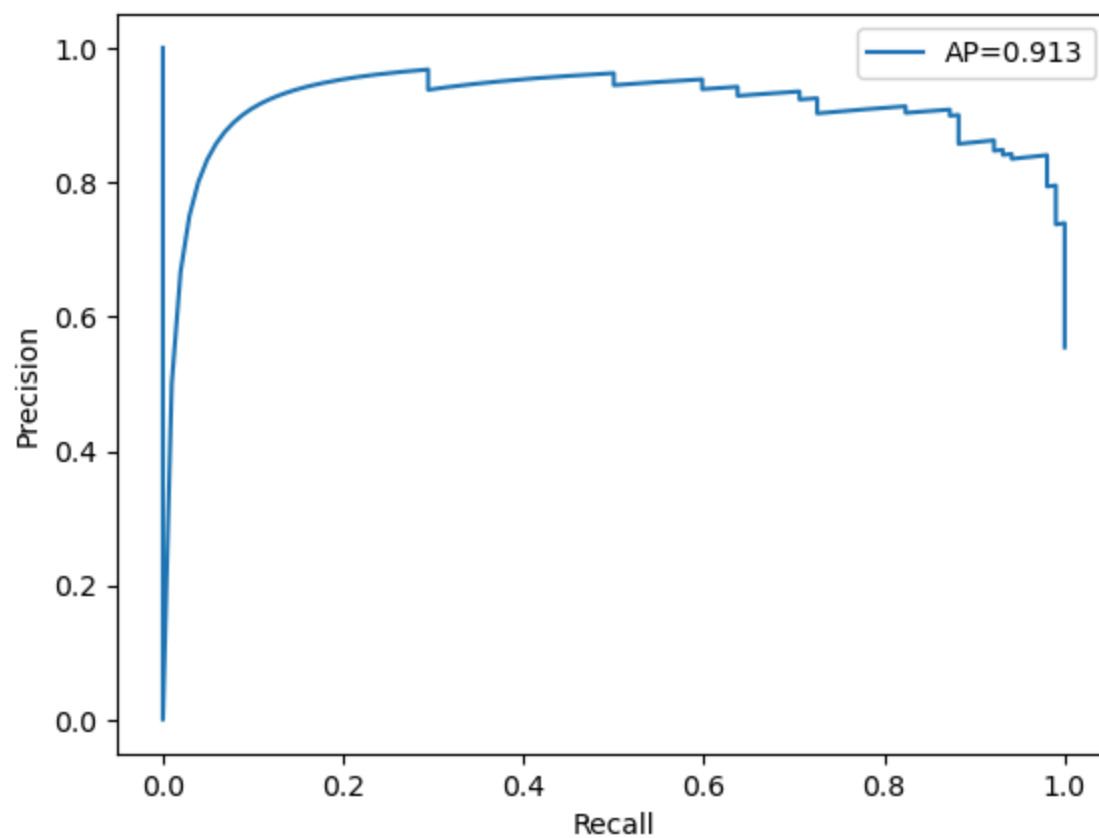
	precision	recall	f1-score	support
0	0.8904	0.7927	0.8387	82
1	0.8468	0.9216	0.8826	102
accuracy			0.8641	184
macro avg	0.8686	0.8571	0.8607	184
weighted avg	0.8663	0.8641	0.8631	184



ROC - Baseline - SVM

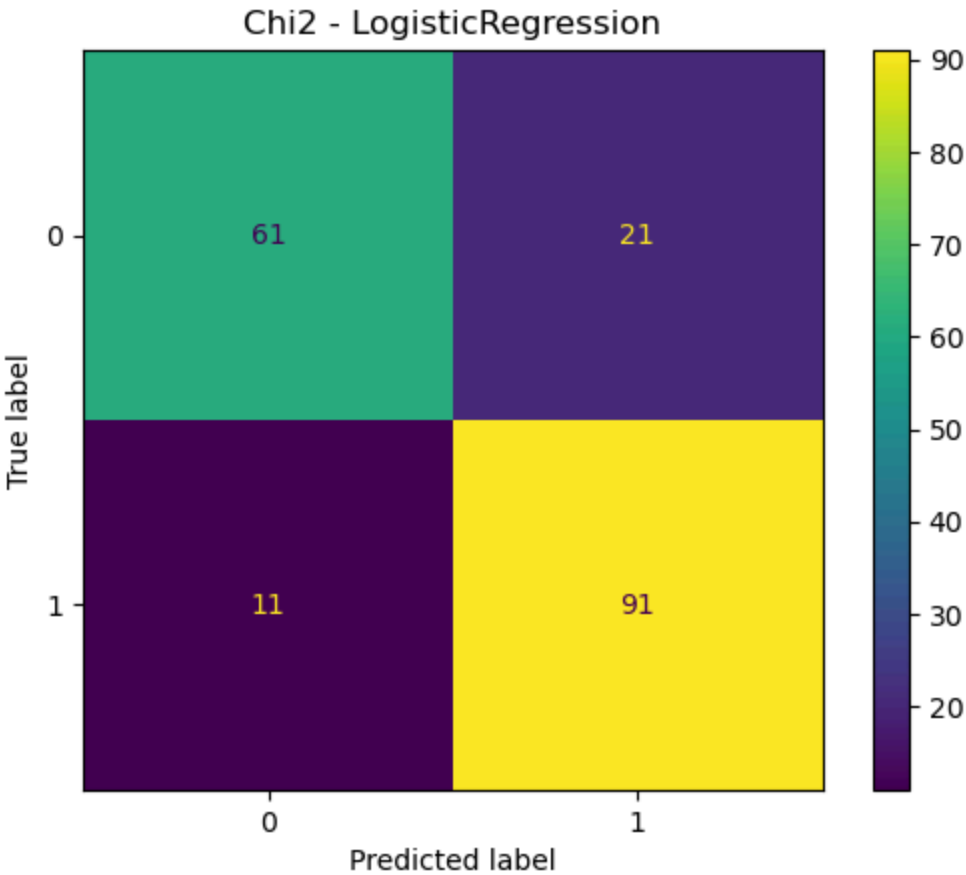


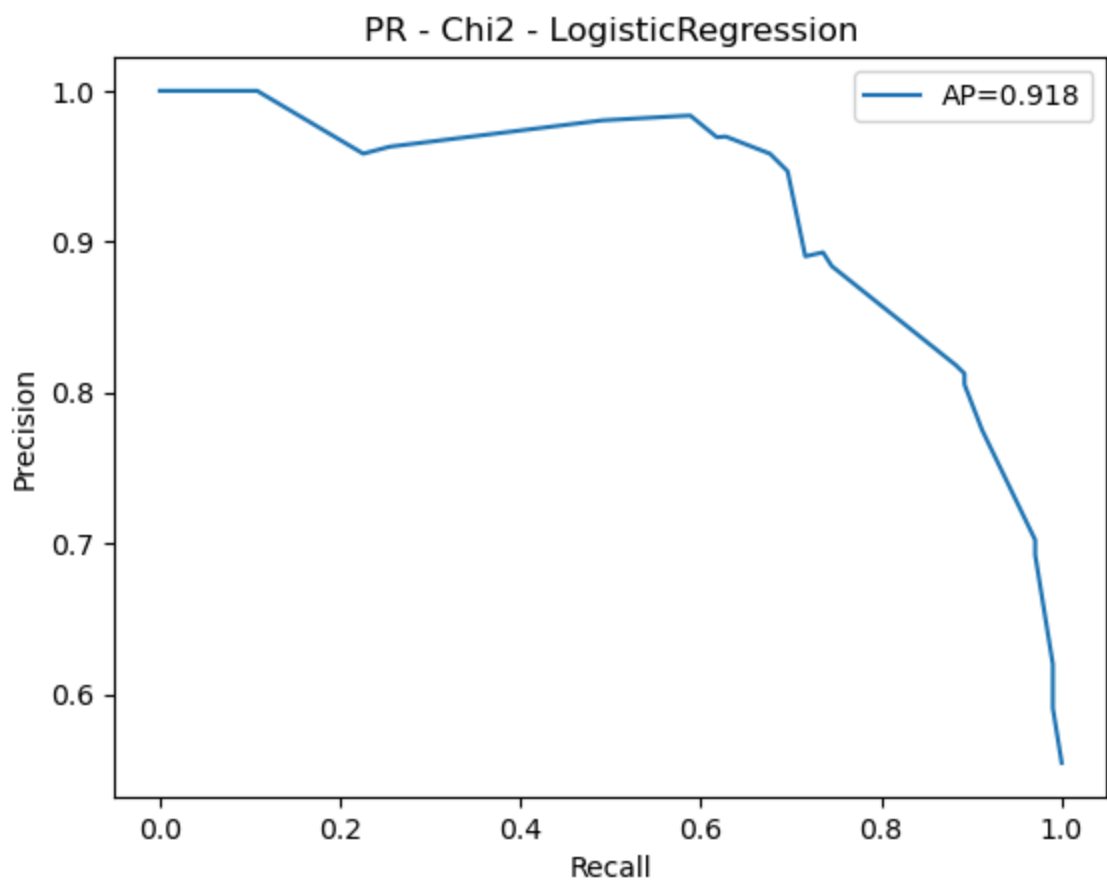
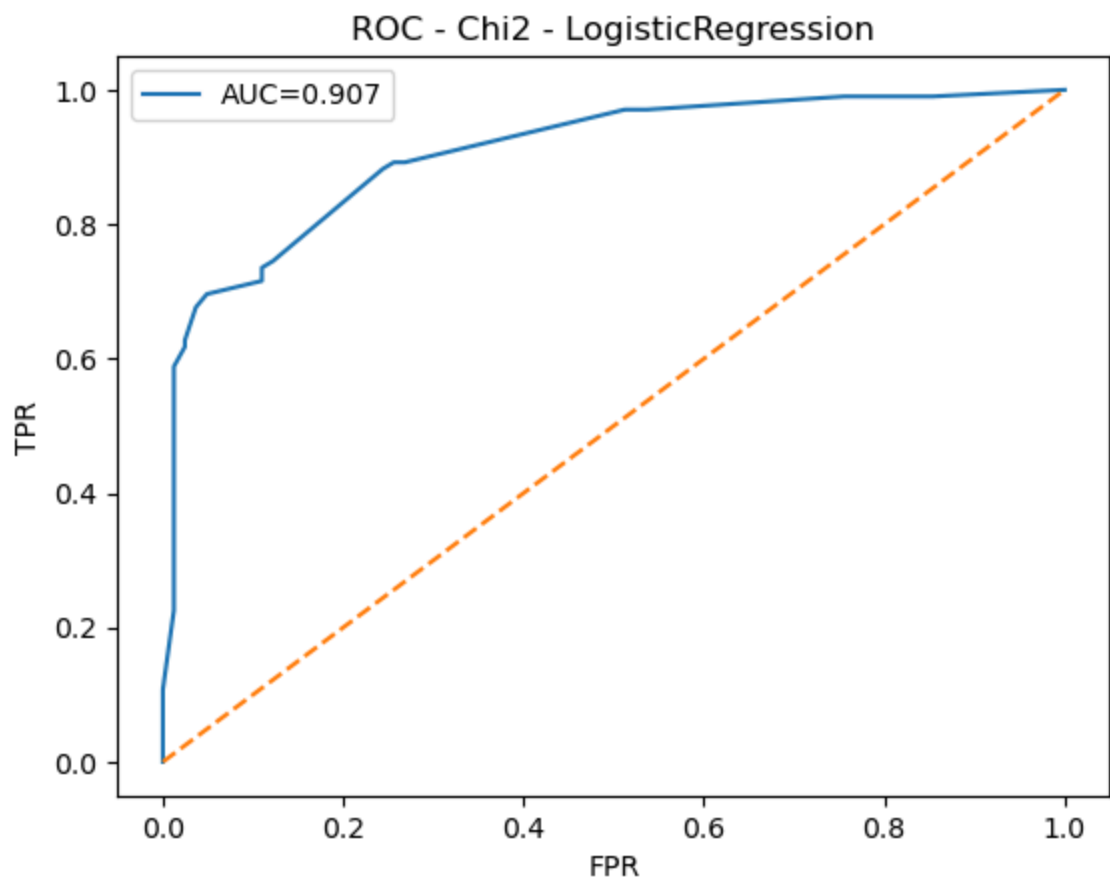
PR - Baseline - SVM



=== FEATURE SELECTION: SelectKBest(chi2) ===

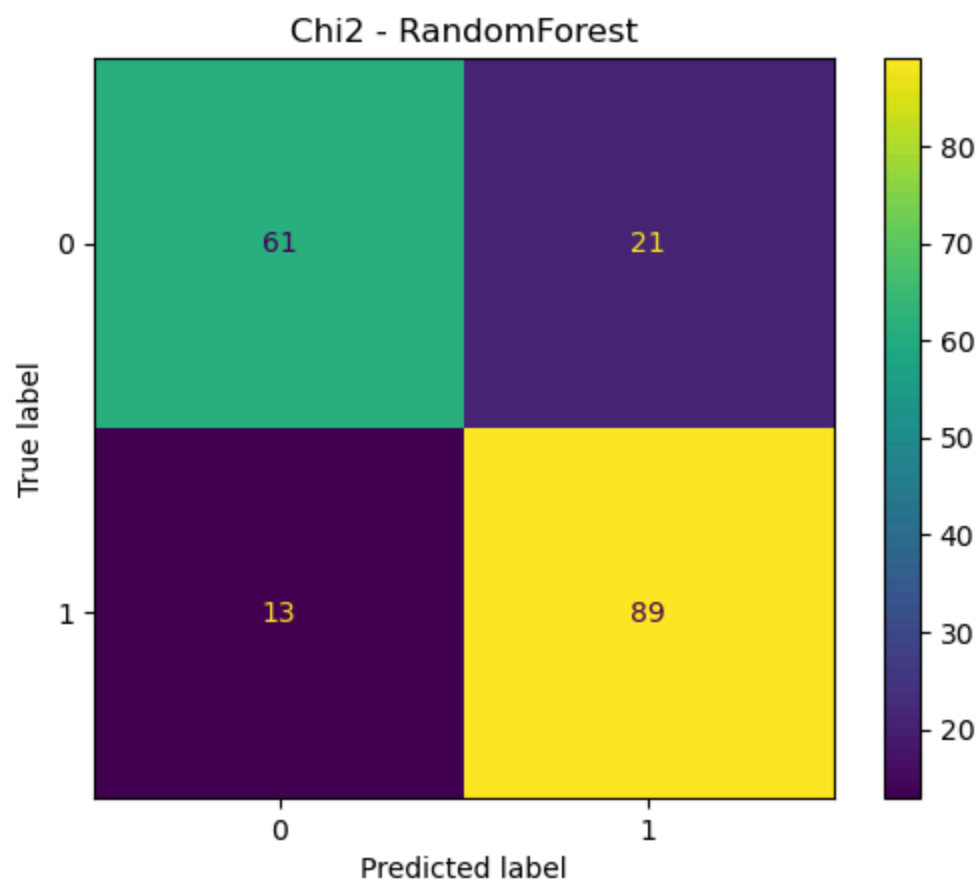
Chi2 - LogisticRegression					
	precision	recall	f1-score	support	
0	0.8472	0.7439	0.7922	82	
1	0.8125	0.8922	0.8505	102	
accuracy			0.8261	184	
macro avg	0.8299	0.8180	0.8213	184	
weighted avg	0.8280	0.8261	0.8245	184	

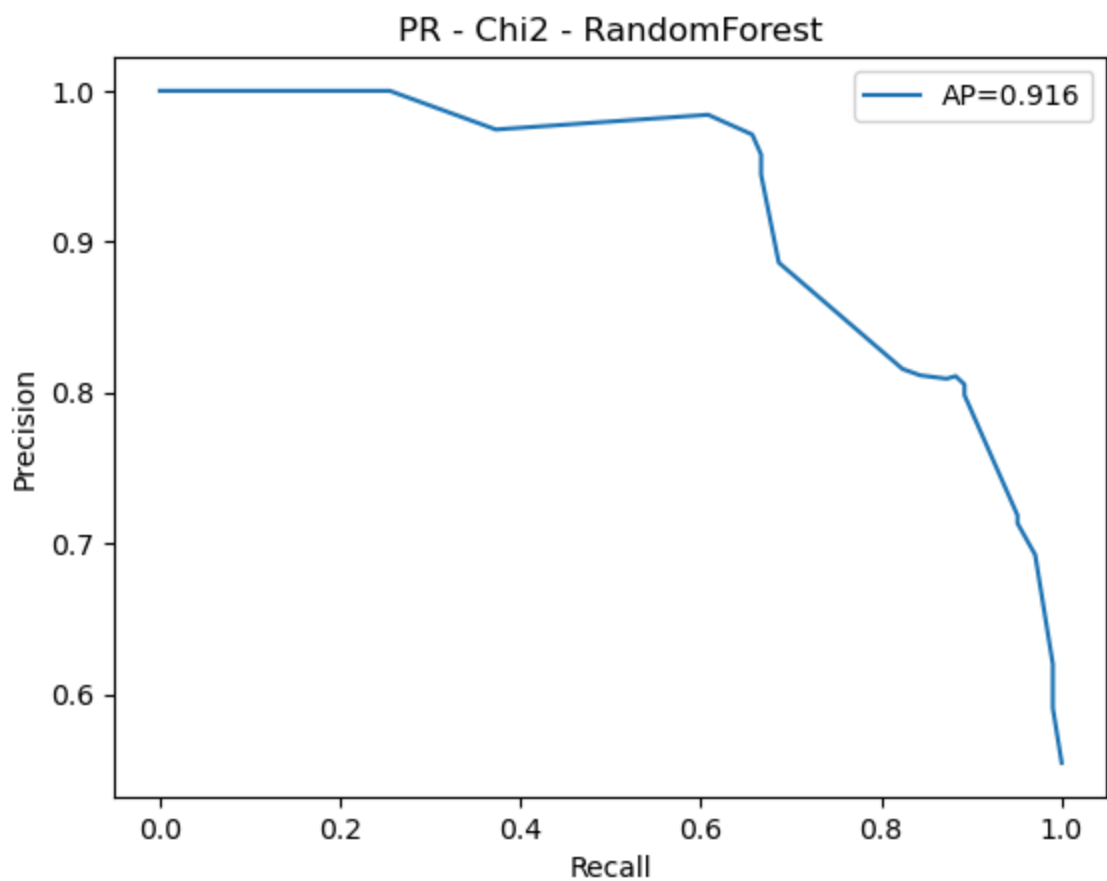
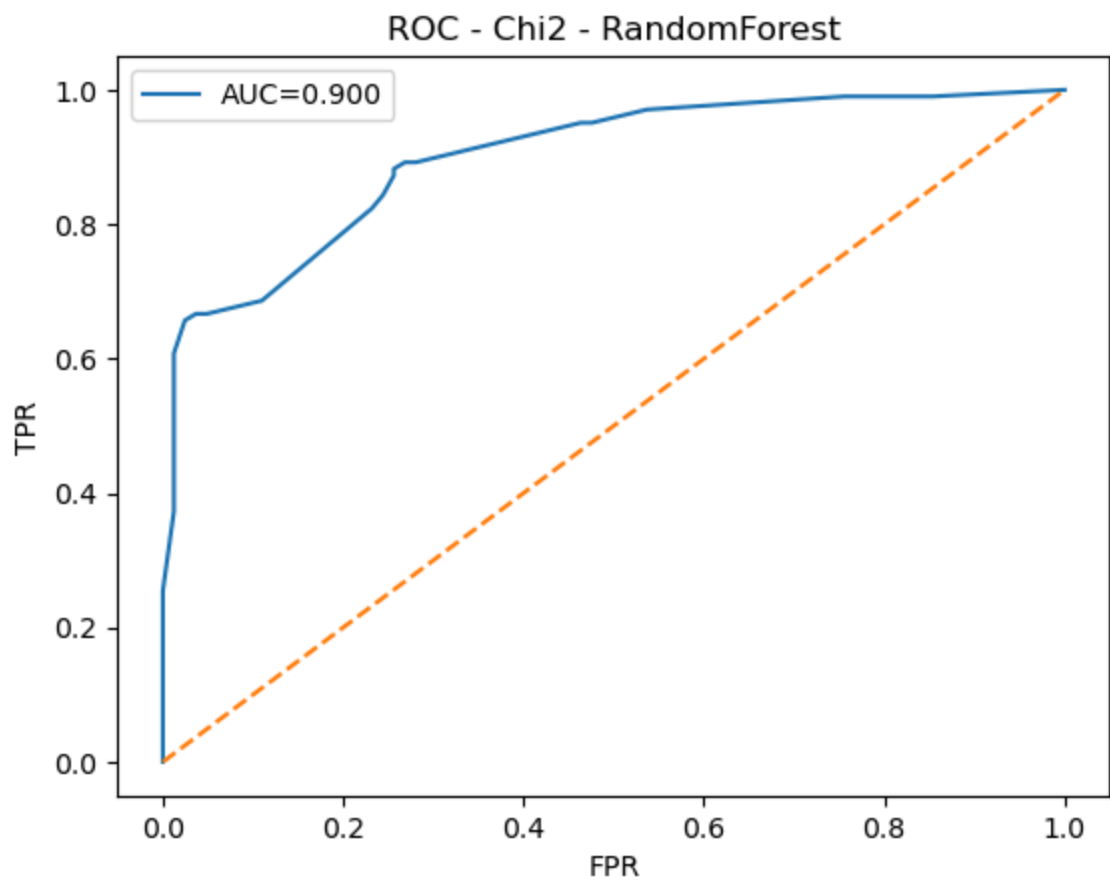




Chi2 - RandomForest

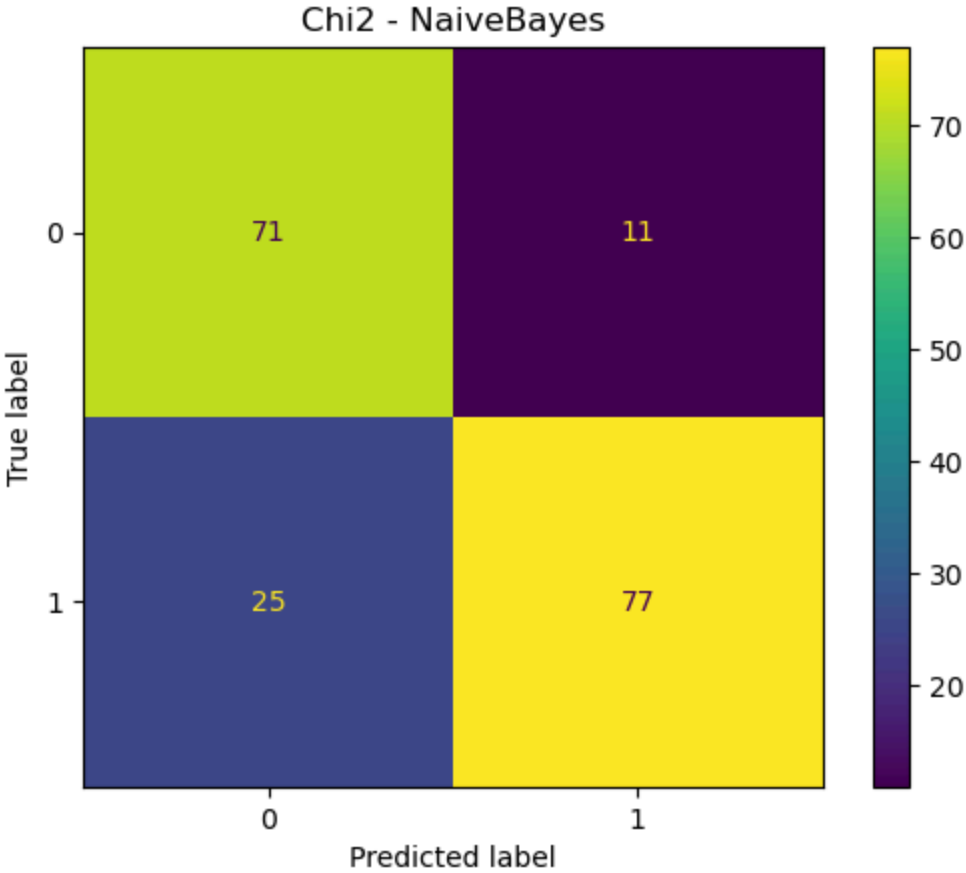
	precision	recall	f1-score	support
0	0.8243	0.7439	0.7821	82
1	0.8091	0.8725	0.8396	102
accuracy			0.8152	184
macro avg	0.8167	0.8082	0.8108	184
weighted avg	0.8159	0.8152	0.8140	184

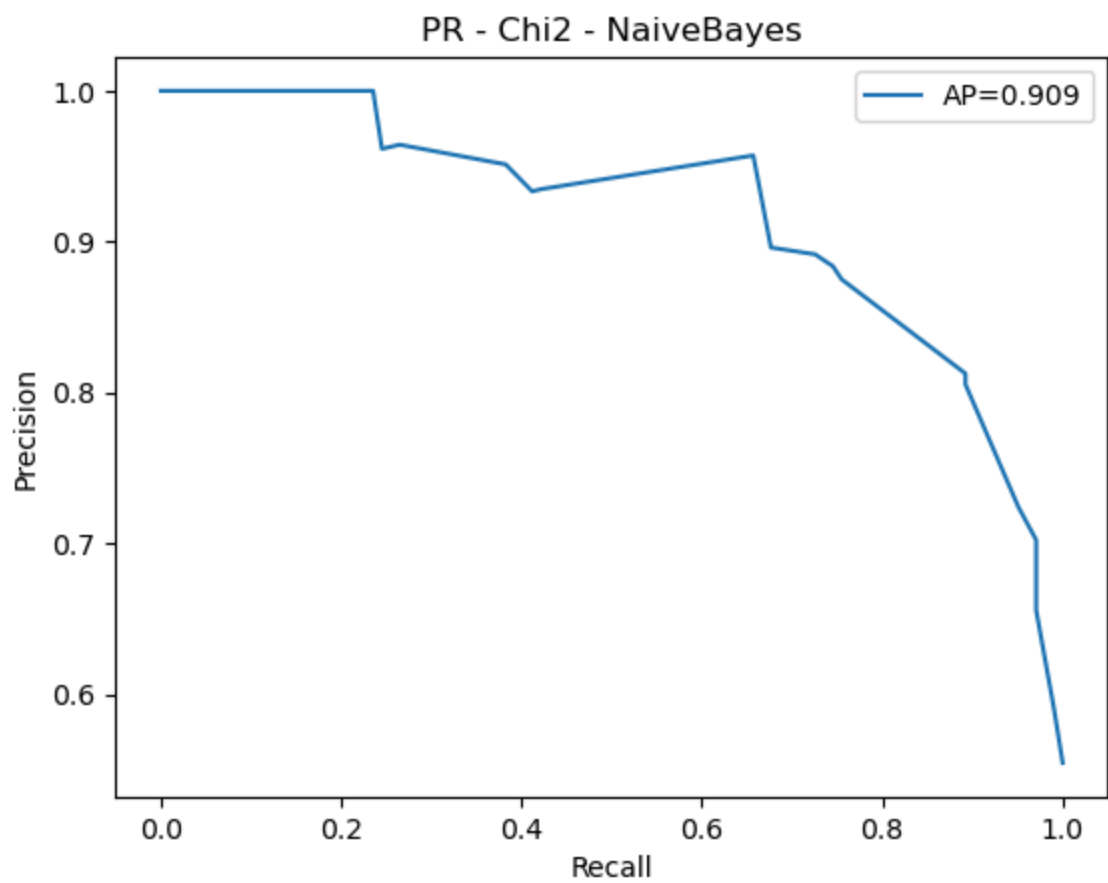
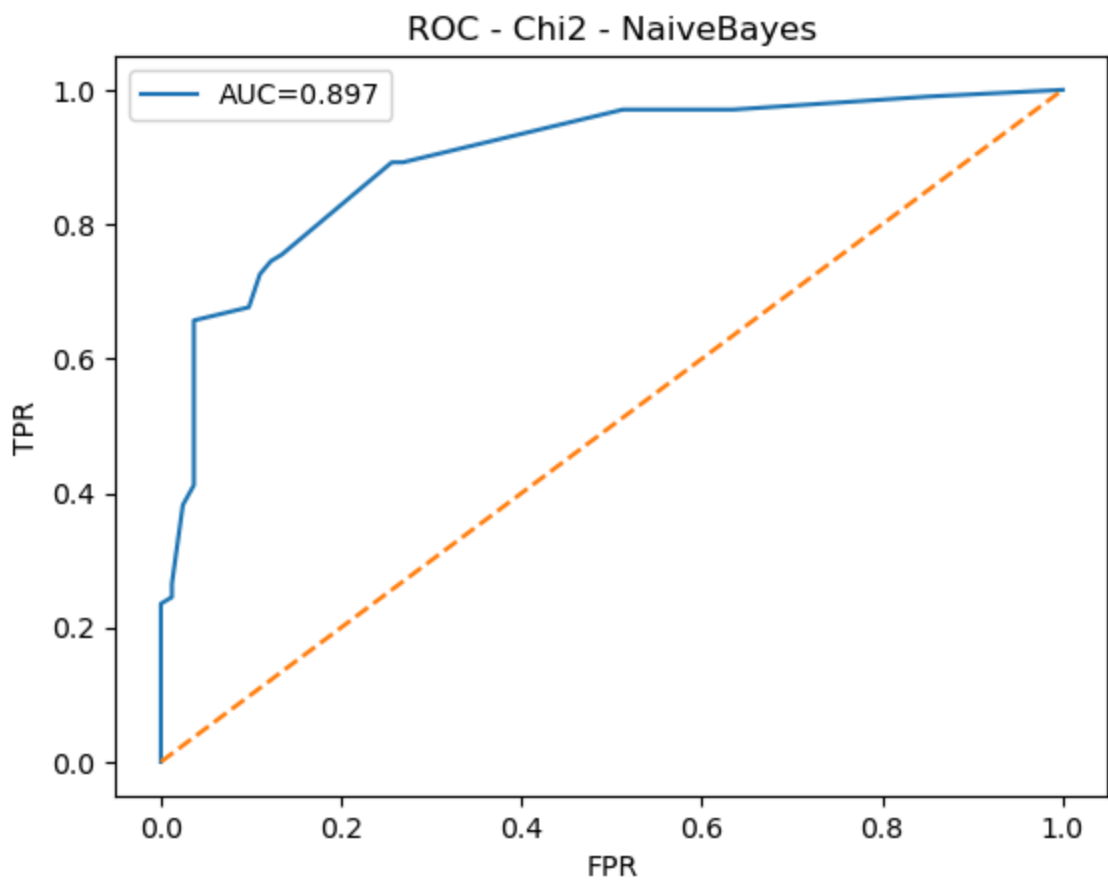




Chi2 - NaiveBayes

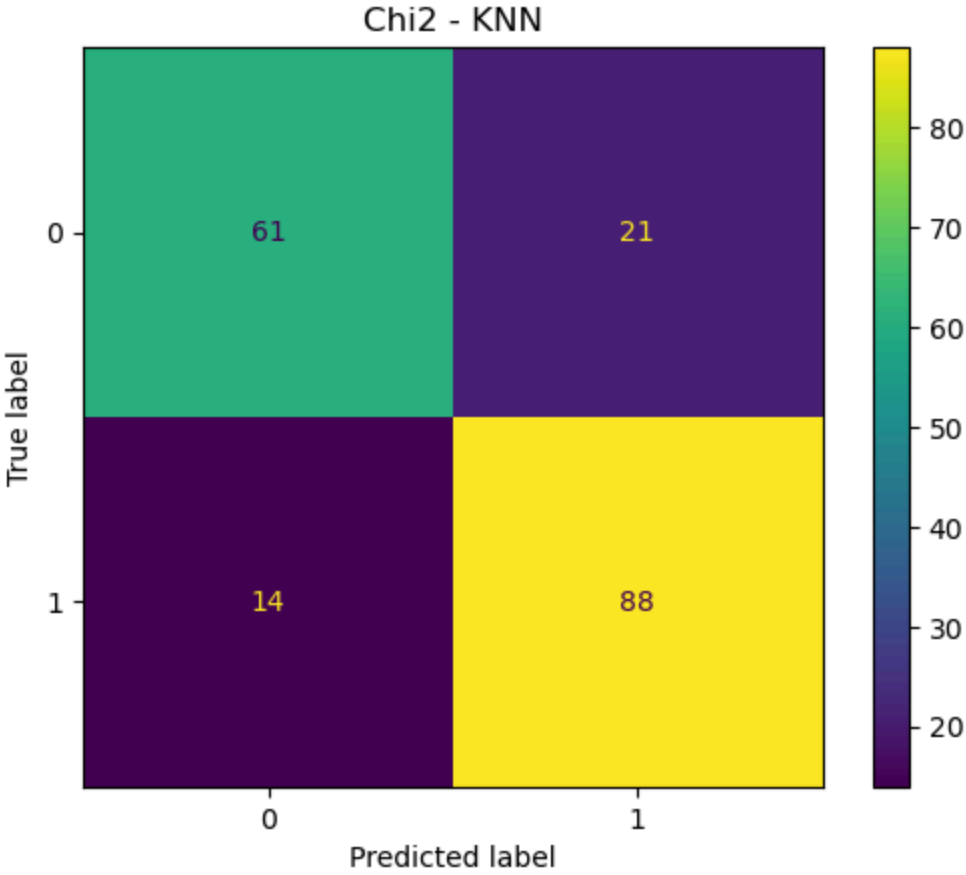
	precision	recall	f1-score	support
0	0.7396	0.8659	0.7978	82
1	0.8750	0.7549	0.8105	102
accuracy			0.8043	184
macro avg	0.8073	0.8104	0.8041	184
weighted avg	0.8147	0.8043	0.8048	184



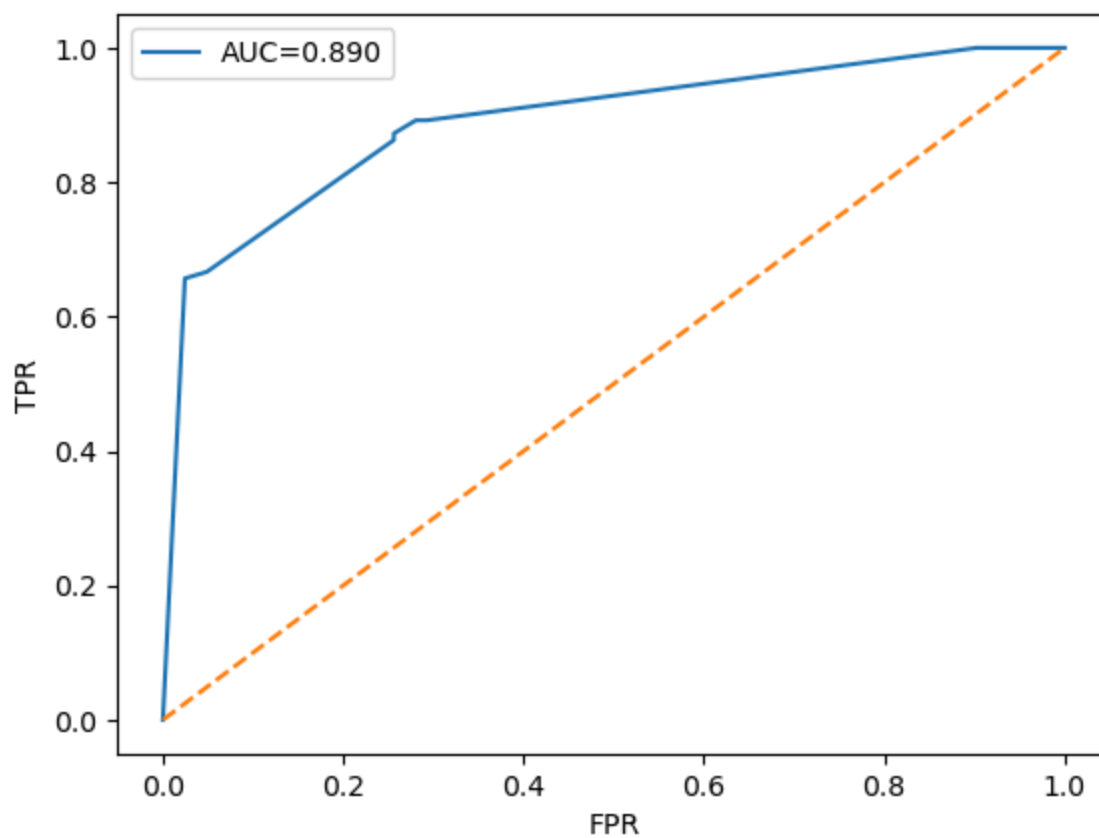


Chi2 - KNN

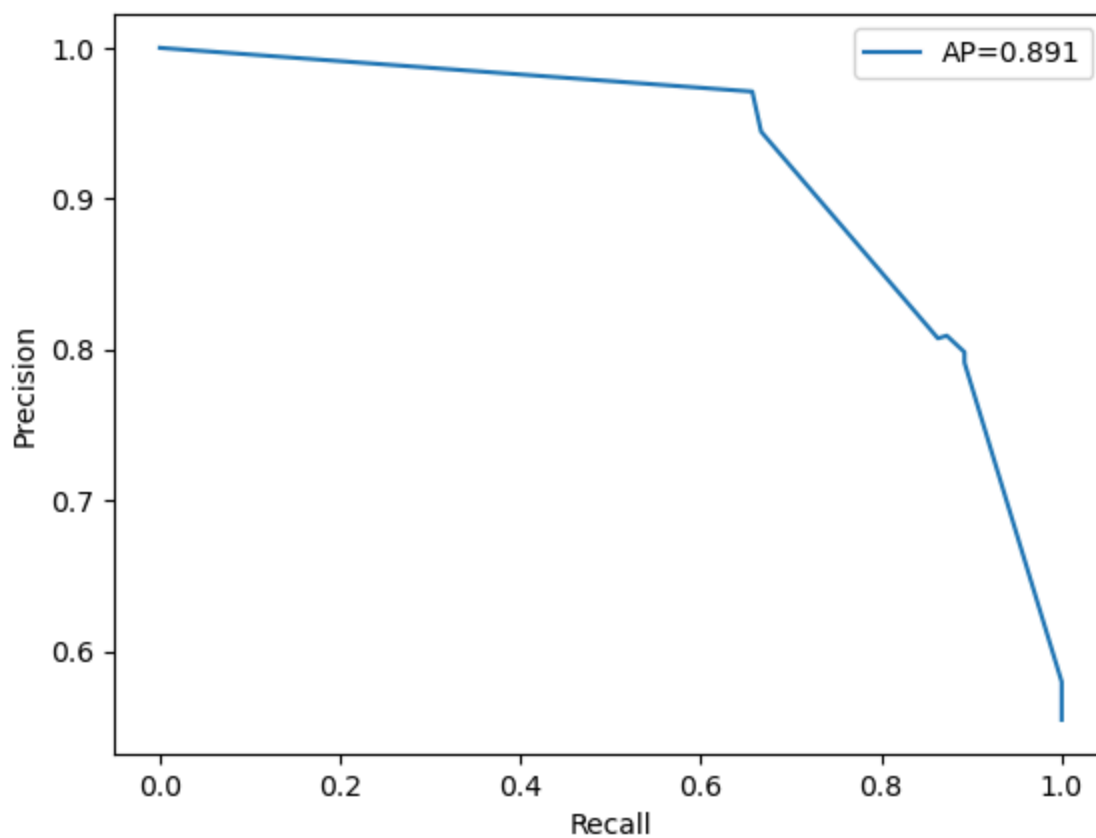
	precision	recall	f1-score	support
0	0.8133	0.7439	0.7771	82
1	0.8073	0.8627	0.8341	102
accuracy			0.8098	184
macro avg	0.8103	0.8033	0.8056	184
weighted avg	0.8100	0.8098	0.8087	184



ROC - Chi2 - KNN

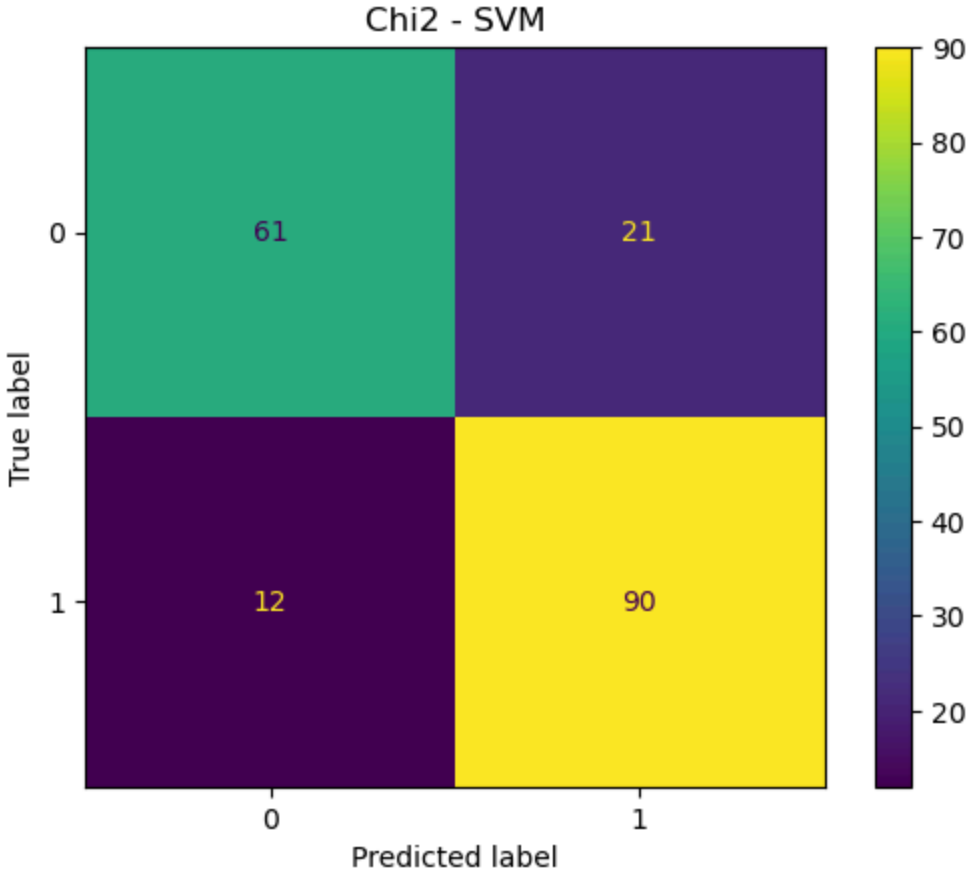


PR - Chi2 - KNN

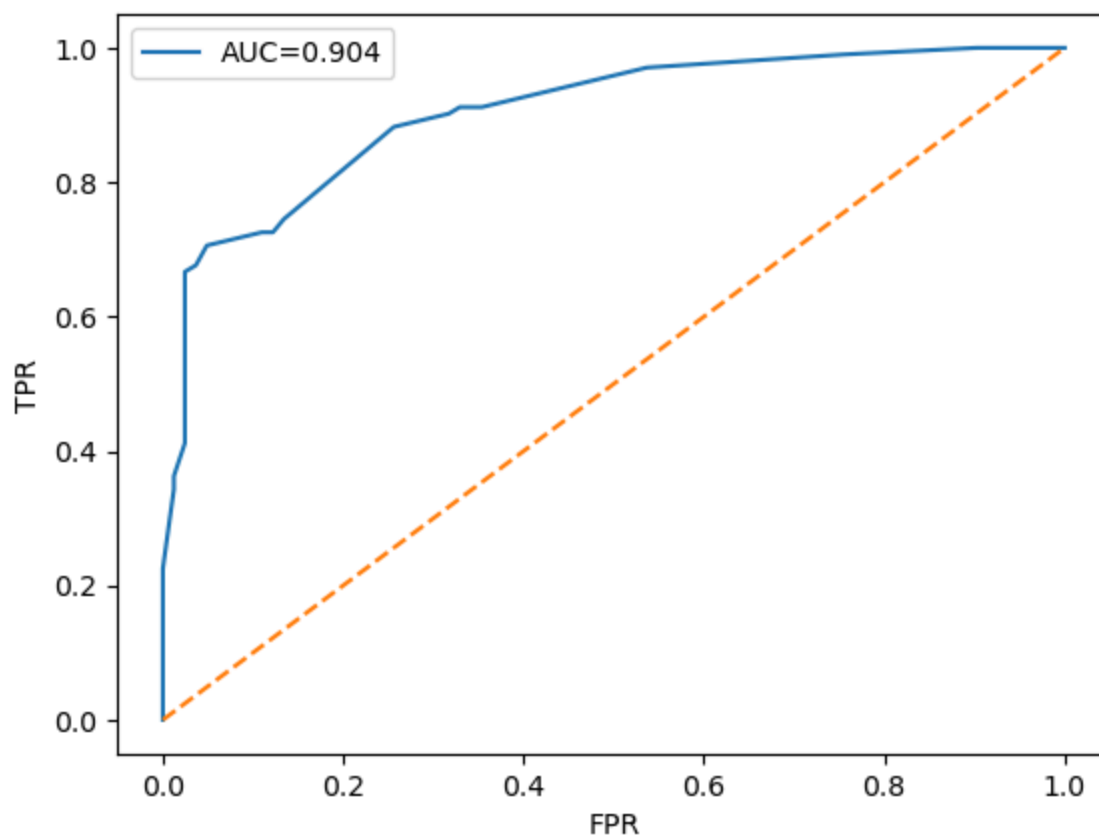


Chi2 - SVM

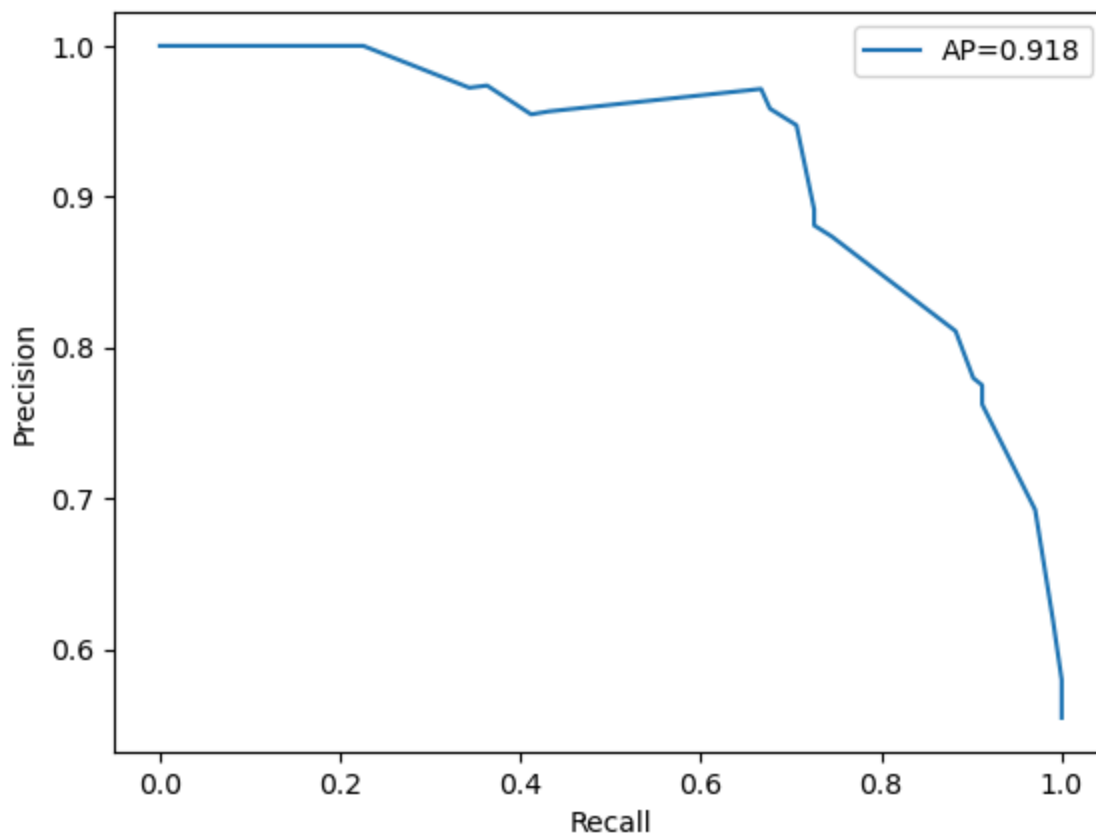
	precision	recall	f1-score	support
0	0.8356	0.7439	0.7871	82
1	0.8108	0.8824	0.8451	102
accuracy			0.8207	184
macro avg	0.8232	0.8131	0.8161	184
weighted avg	0.8219	0.8207	0.8192	184



ROC - Chi2 - SVM

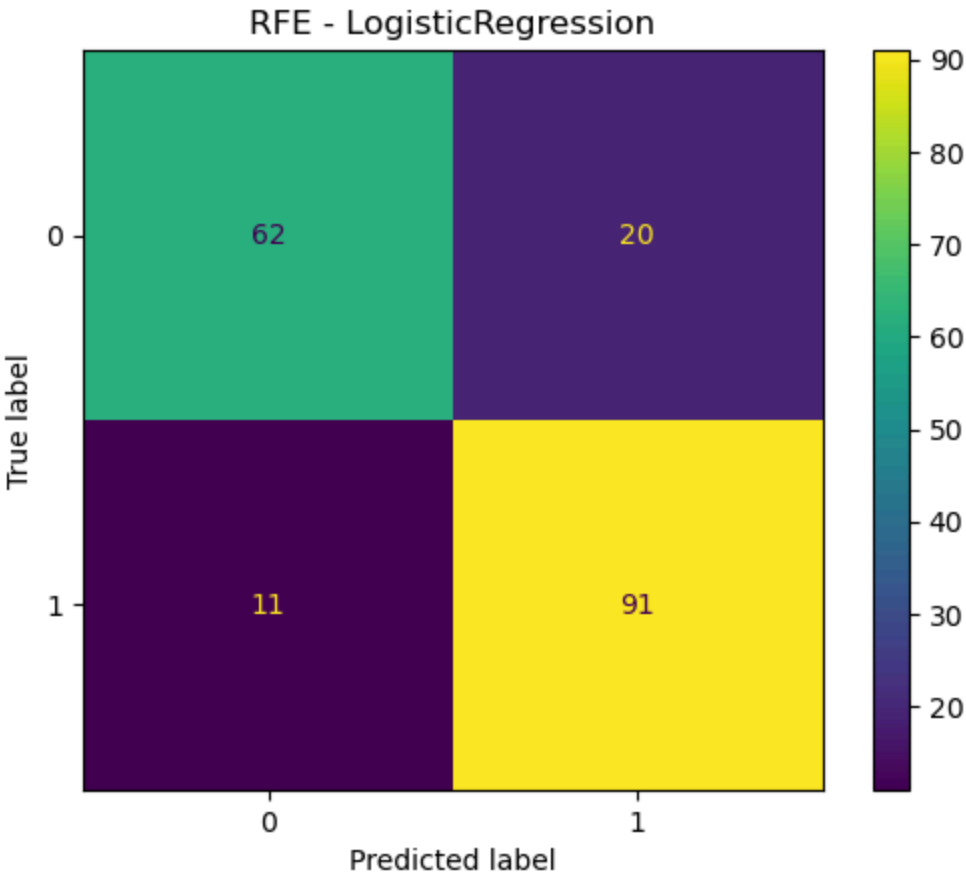


PR - Chi2 - SVM

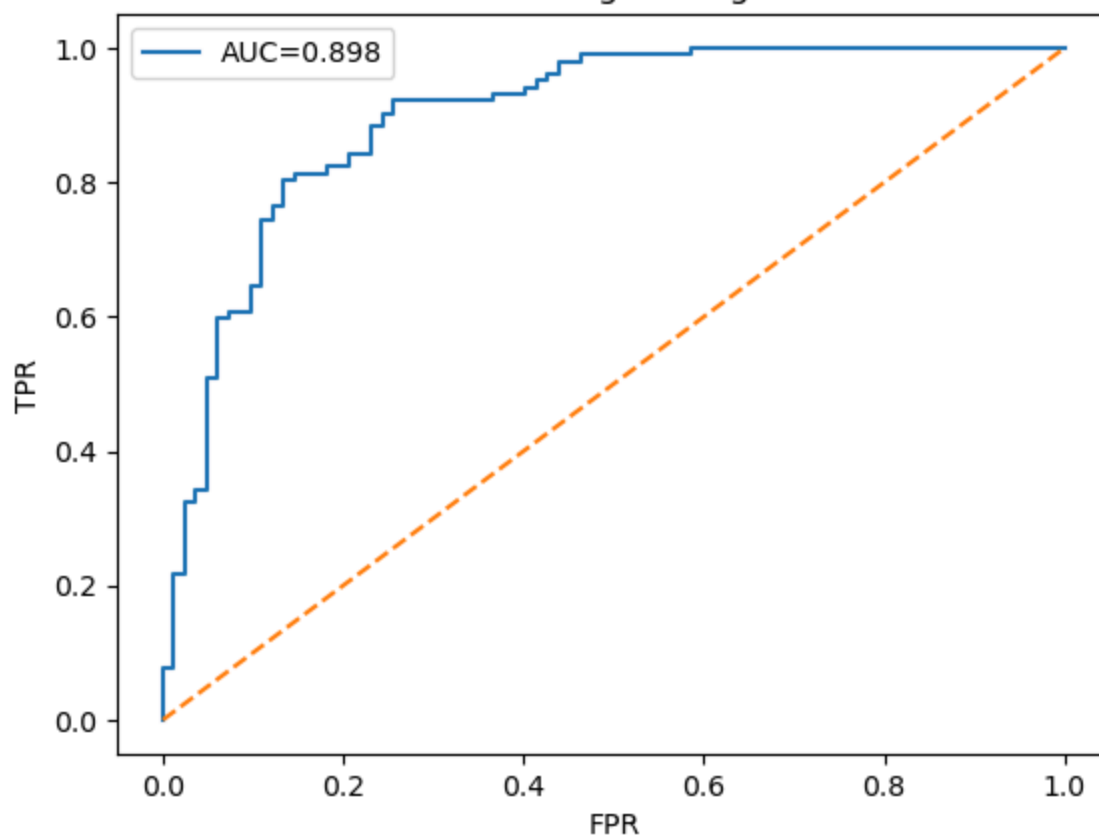


=== FEATURE SELECTION: RFE (with RandomForest) ===

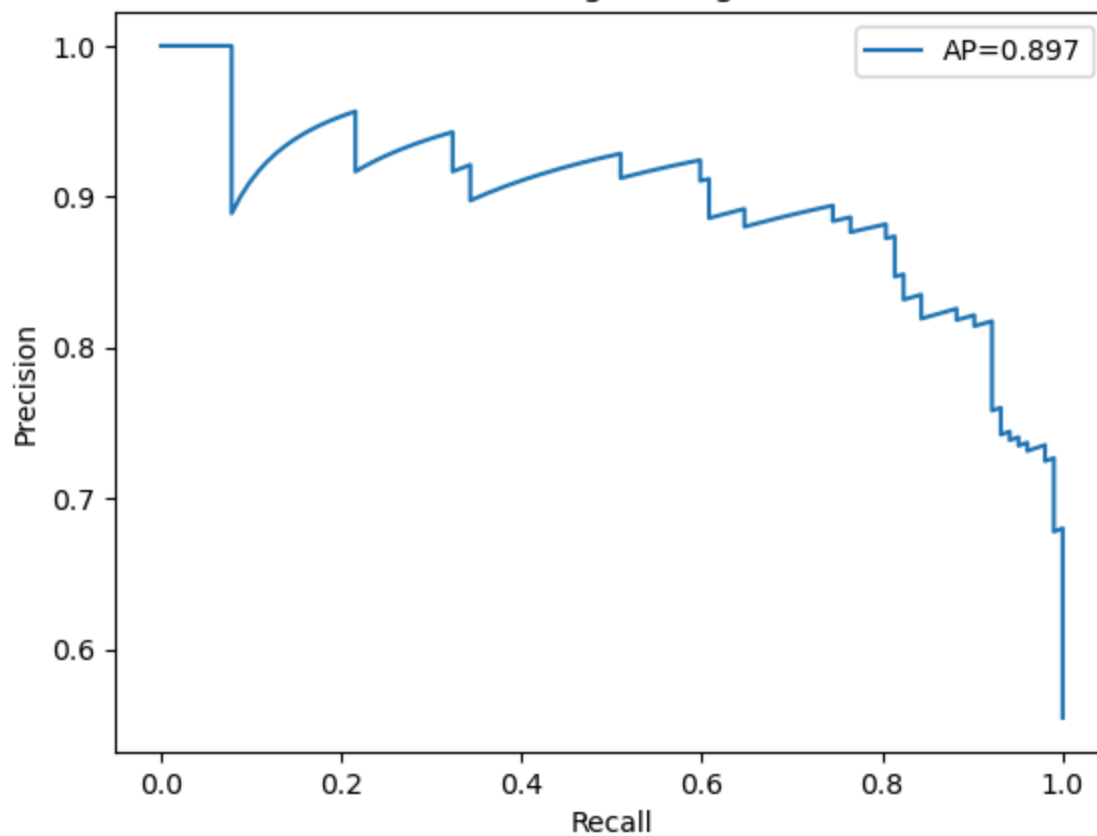
RFE - LogisticRegression					
	precision	recall	f1-score	support	
0	0.8493	0.7561	0.8000	82	
1	0.8198	0.8922	0.8545	102	
accuracy			0.8315	184	
macro avg	0.8346	0.8241	0.8272	184	
weighted avg	0.8330	0.8315	0.8302	184	



ROC - RFE - LogisticRegression

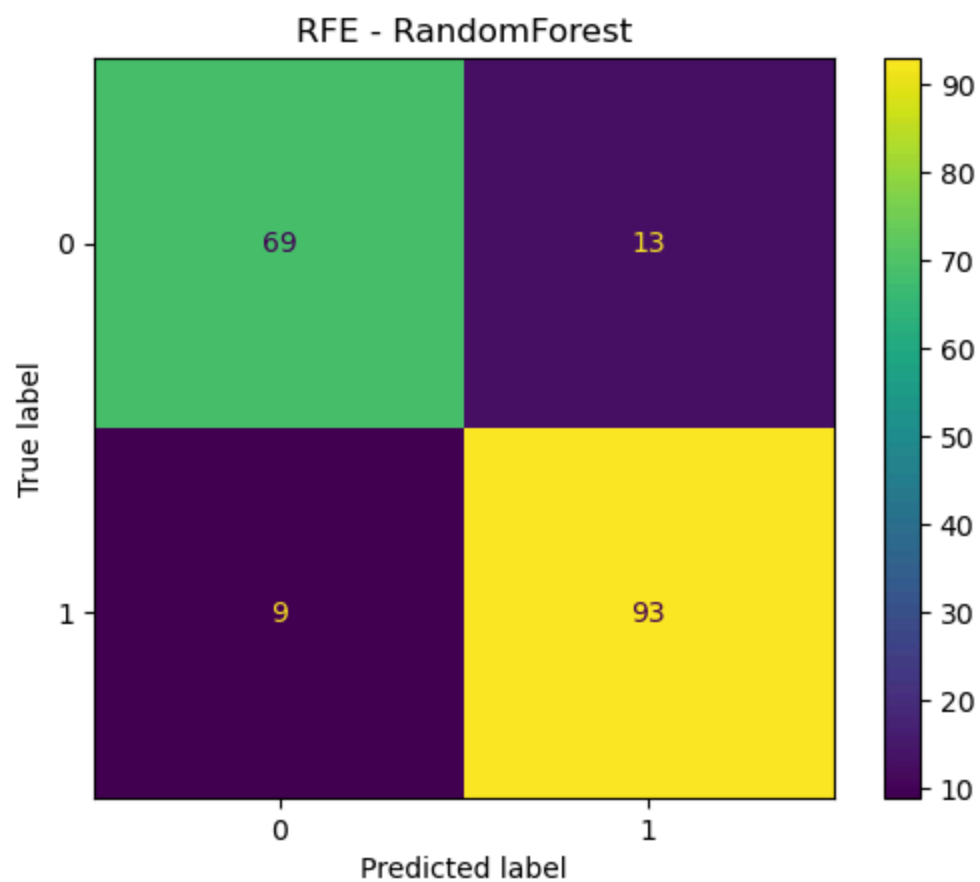


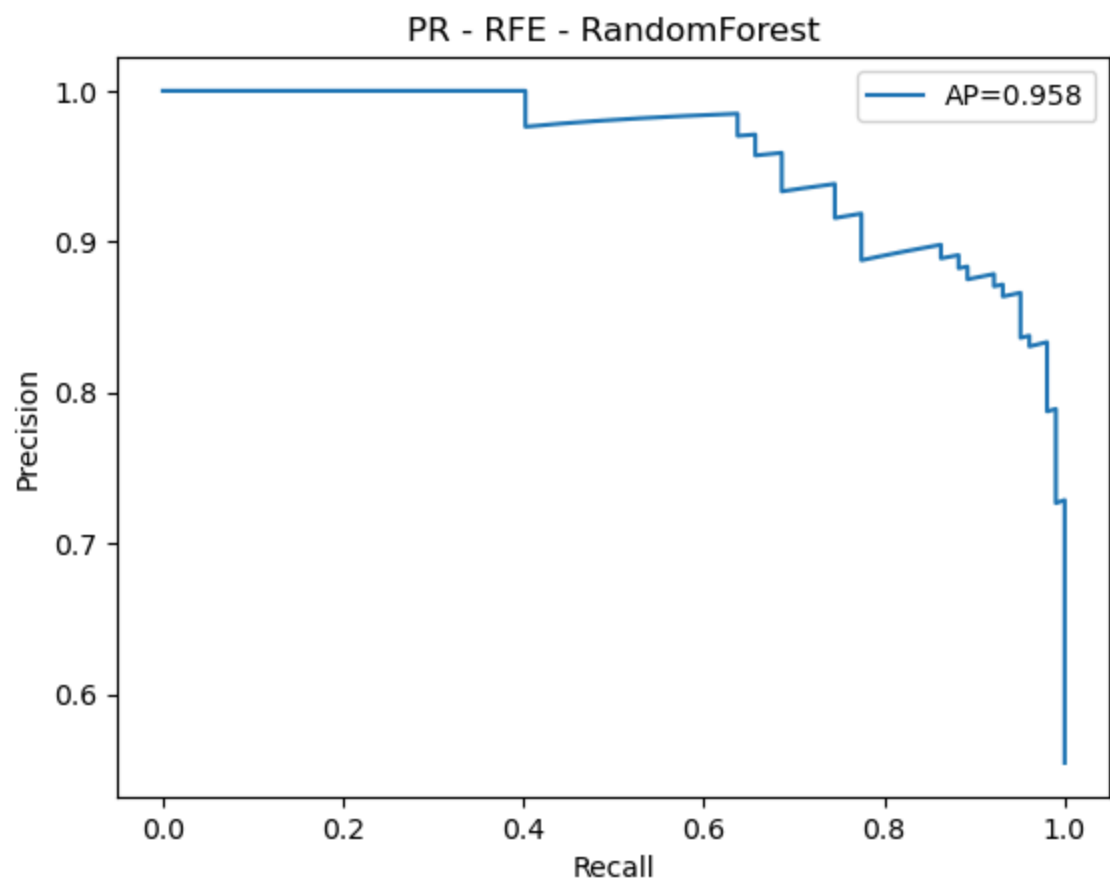
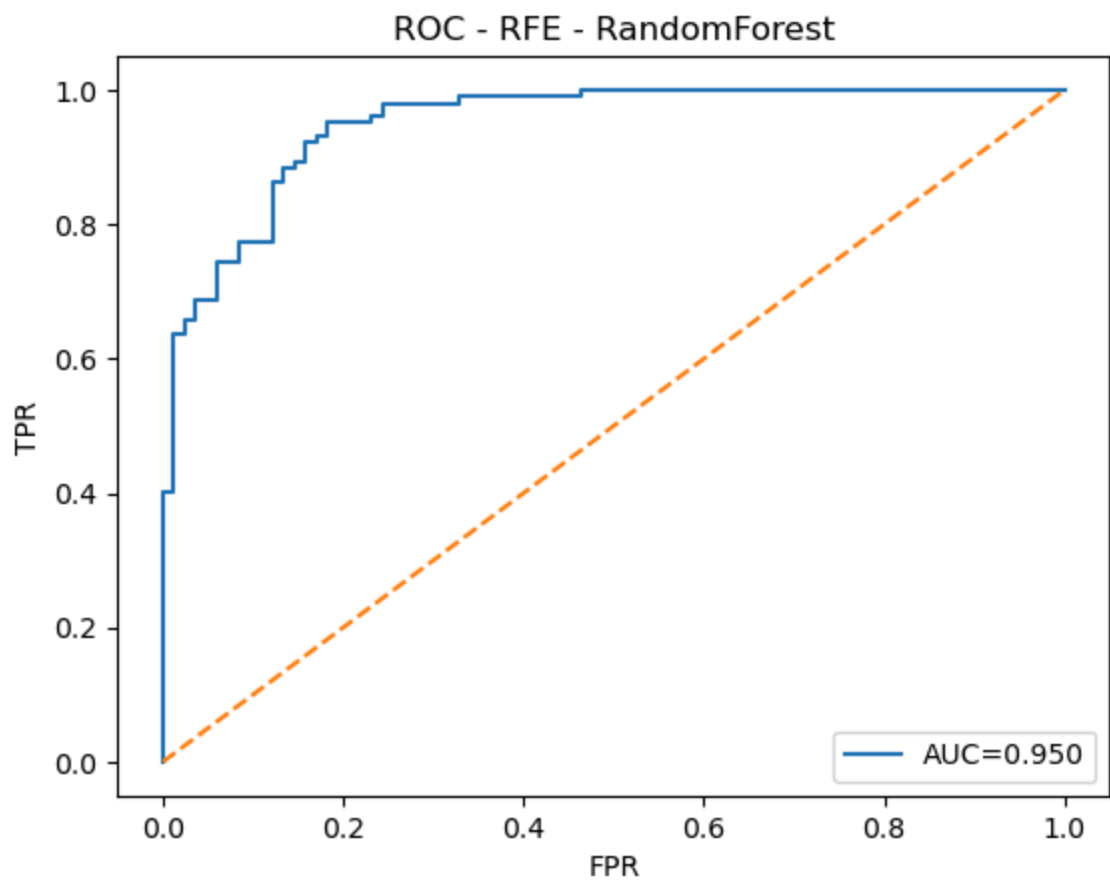
PR - RFE - LogisticRegression



RFE - RandomForest

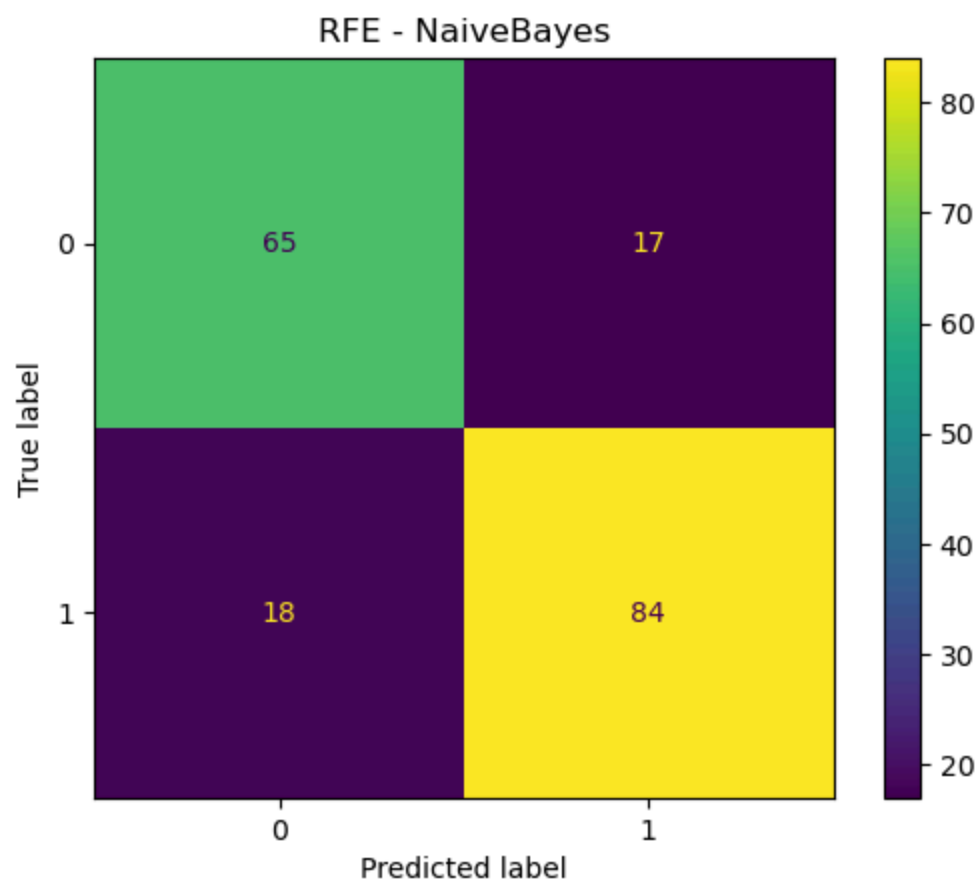
	precision	recall	f1-score	support
0	0.8846	0.8415	0.8625	82
1	0.8774	0.9118	0.8942	102
accuracy			0.8804	184
macro avg	0.8810	0.8766	0.8784	184
weighted avg	0.8806	0.8804	0.8801	184



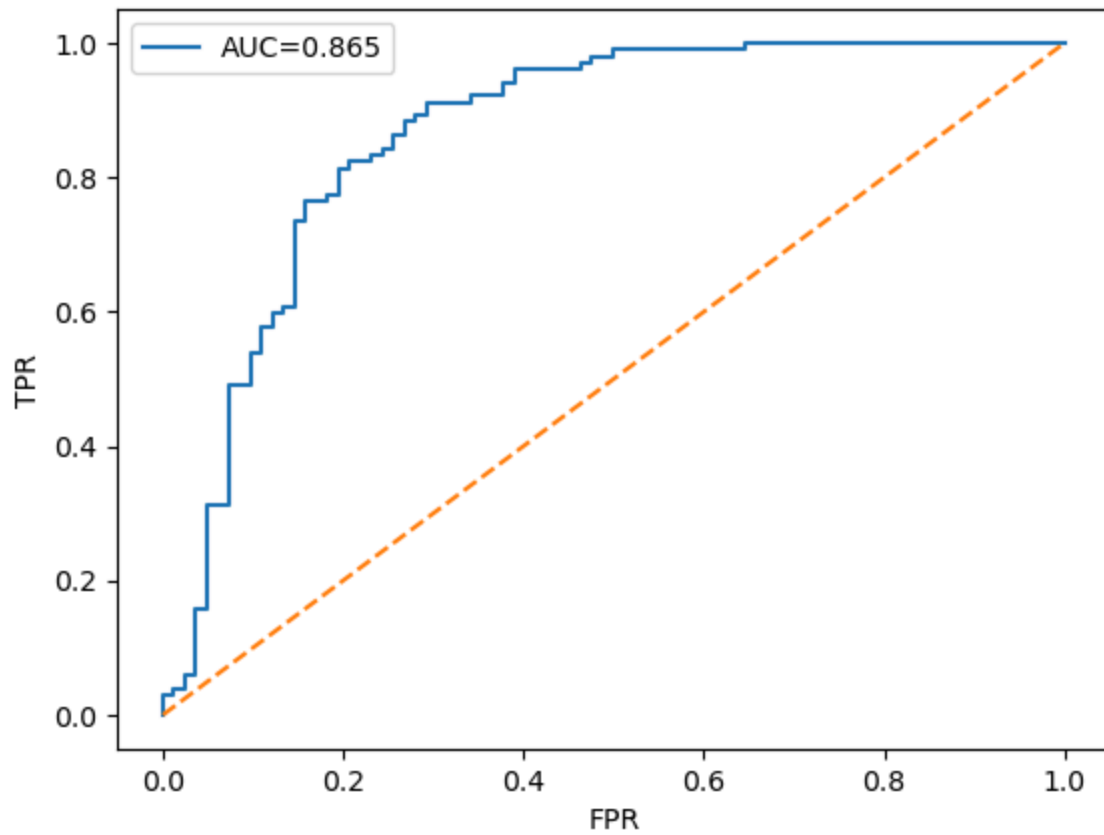


RFE - NaiveBayes

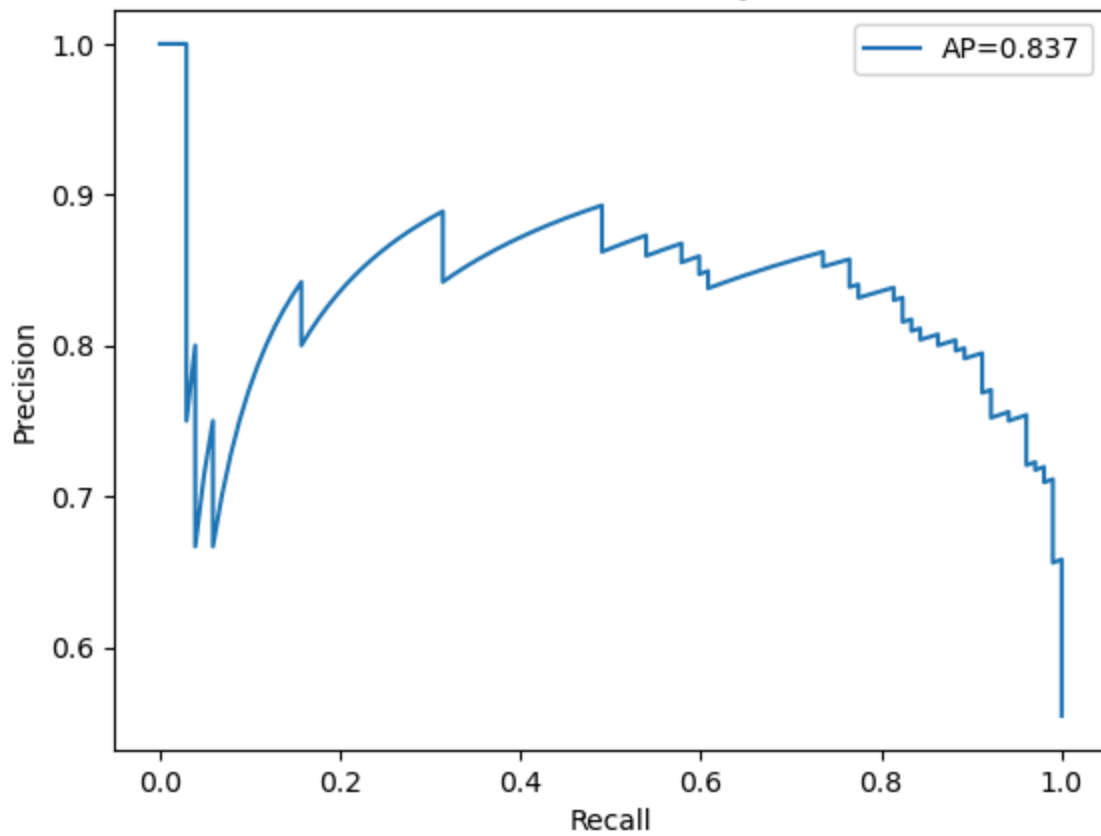
	precision	recall	f1-score	support
0	0.7831	0.7927	0.7879	82
1	0.8317	0.8235	0.8276	102
accuracy			0.8098	184
macro avg	0.8074	0.8081	0.8077	184
weighted avg	0.8100	0.8098	0.8099	184



ROC - RFE - NaiveBayes

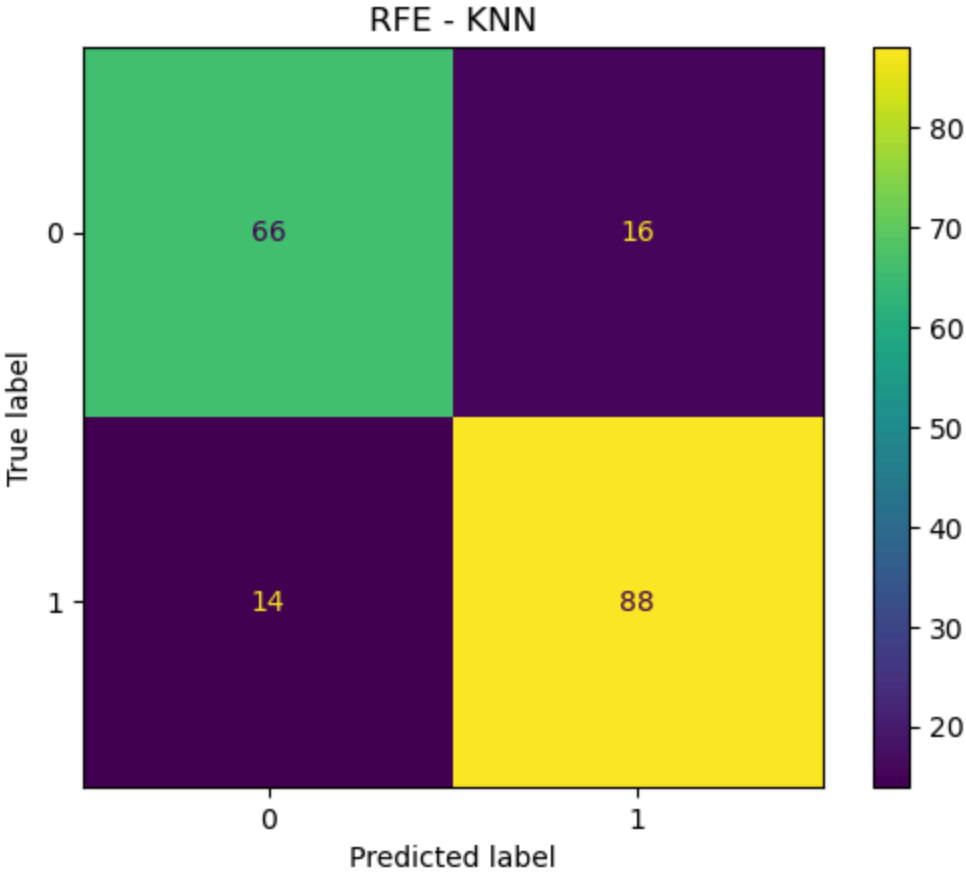


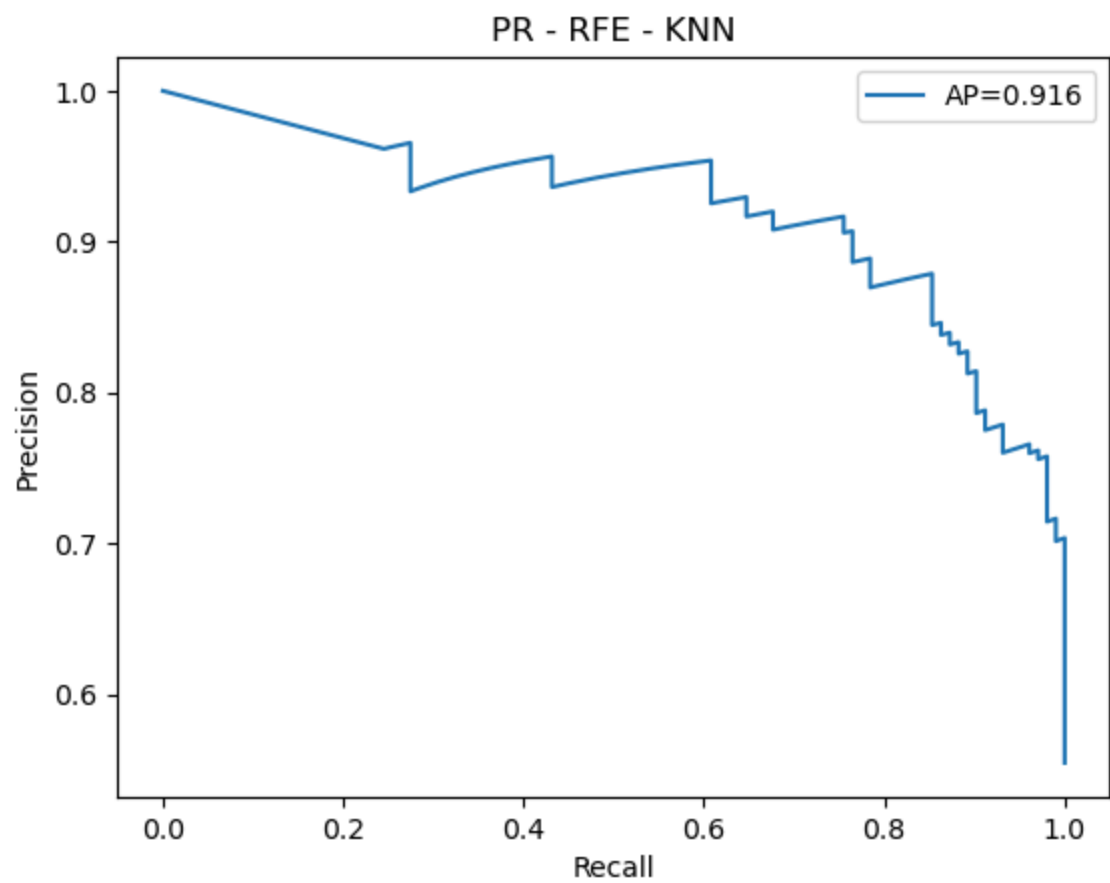
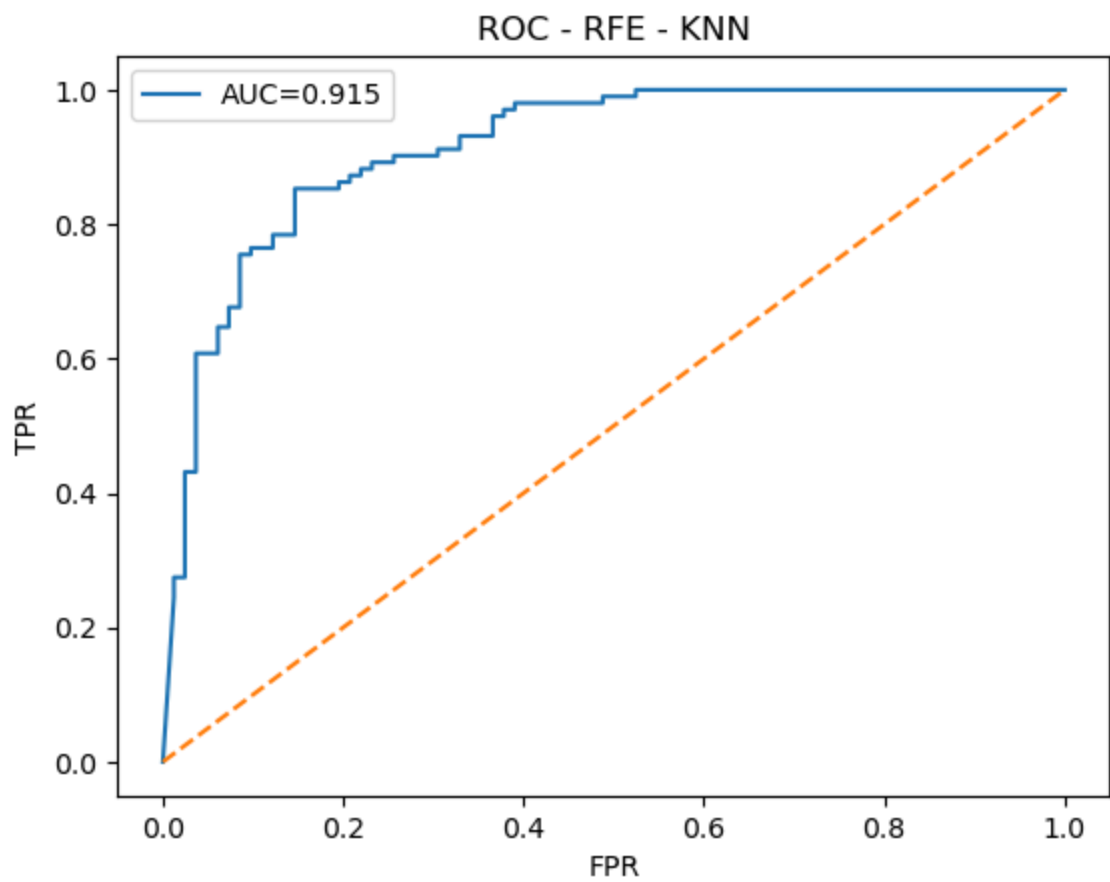
PR - RFE - NaiveBayes



RFE - KNN

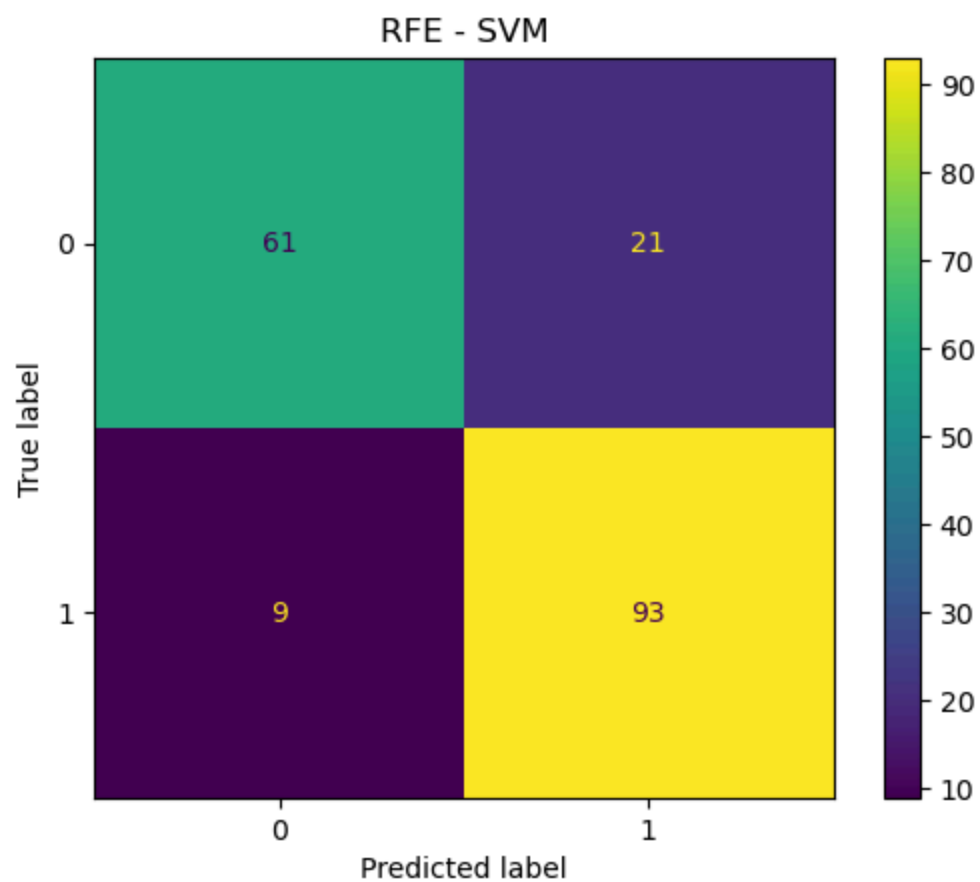
	precision	recall	f1-score	support
0	0.8250	0.8049	0.8148	82
1	0.8462	0.8627	0.8544	102
accuracy			0.8370	184
macro avg	0.8356	0.8338	0.8346	184
weighted avg	0.8367	0.8370	0.8367	184

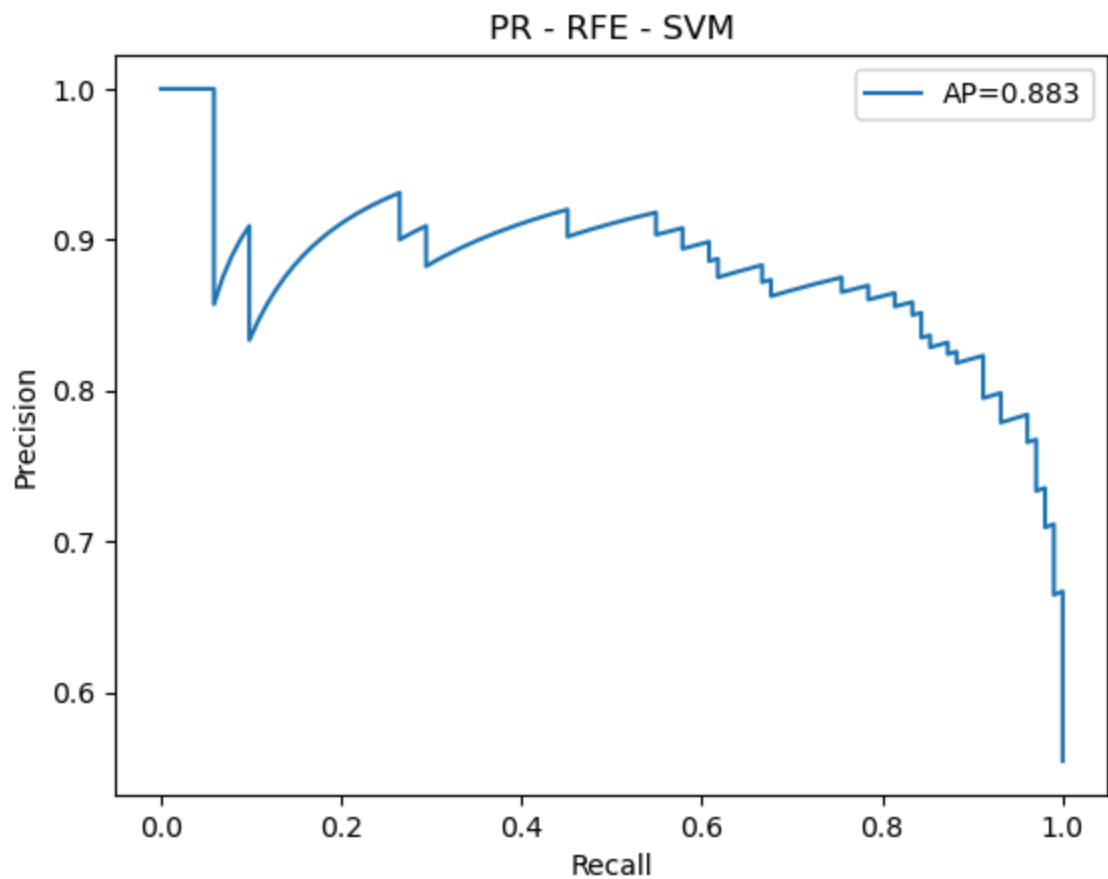
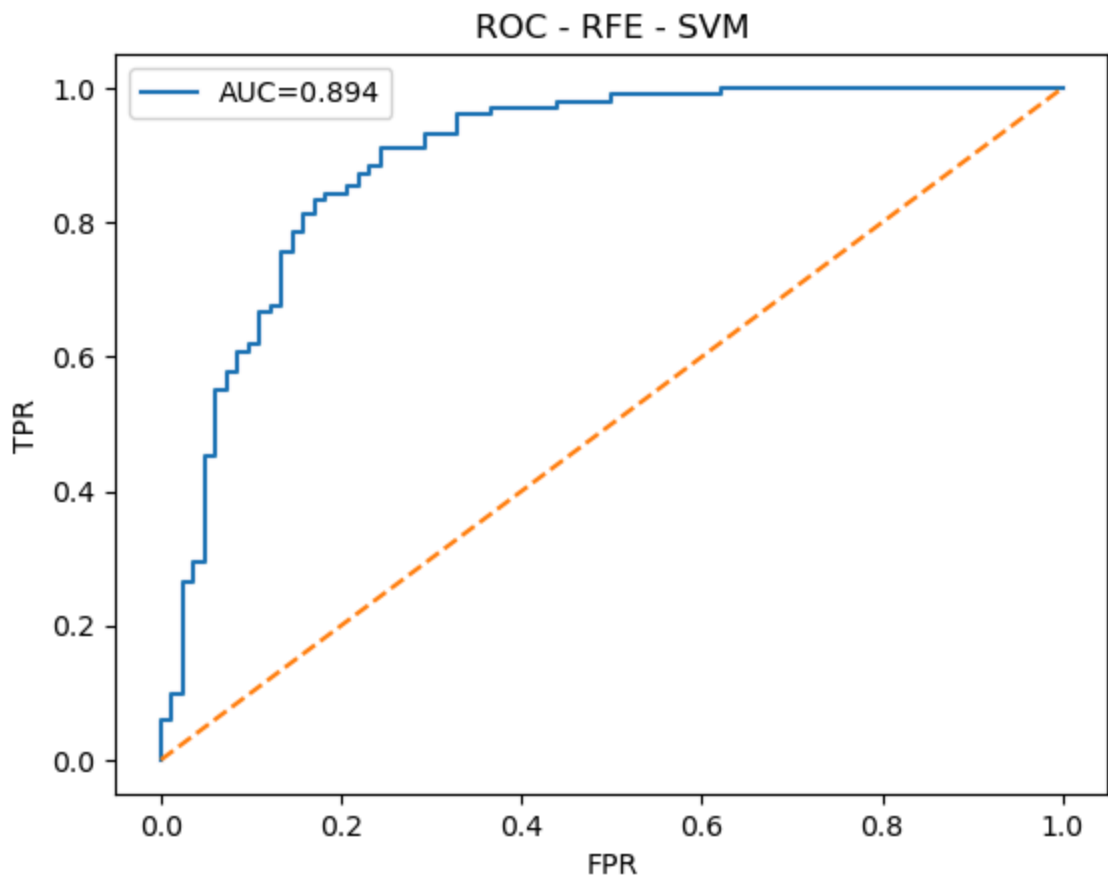




RFE - SVM

	precision	recall	f1-score	support
0	0.8714	0.7439	0.8026	82
1	0.8158	0.9118	0.8611	102
accuracy			0.8370	184
macro avg	0.8436	0.8278	0.8319	184
weighted avg	0.8406	0.8370	0.8350	184





Saved summary to heart_results_summary.csv

	phase	model	cv_f1
1	baseline	RandomForest	0.891706
0	baseline	LogisticRegression	0.864424
4	baseline	SVM	0.856846
3	baseline	KNN	0.841707
2	baseline	NaiveBayes	0.835243
9	fs_chi2	SVM	0.809875
5	fs_chi2	LogisticRegression	0.801213
8	fs_chi2	KNN	0.800695
6	fs_chi2	RandomForest	0.794862
7	fs_chi2	NaiveBayes	0.786511
11	fs_rfe	RandomForest	0.860563
13	fs_rfe	KNN	0.830837
14	fs_rfe	SVM	0.827750
12	fs_rfe	NaiveBayes	0.827014
10	fs_rfe	LogisticRegression	0.814809