

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.metrics import accuracy_score, classification_report
```

```
import pandas as pd

train_path = "/content/drive/MyDrive/final project/train.csv"
test_path = "/content/drive/MyDrive/final project/test.csv"

train_df = pd.read_csv(train_path)
test_df = pd.read_csv(test_path)

train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85	C	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	

Next steps: [Generate code with train_df](#) [New interactive sheet](#)

```
train_df.info()
train_df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   PassengerId      891 non-null    int64
1   Survived         891 non-null    int64
2   Pclass           891 non-null    int64
3   Name             891 non-null    object
4   Sex              891 non-null    object
5   Age              714 non-null    float64
6   SibSp            891 non-null    int64
7   Parch            891 non-null    int64
8   Ticket           891 non-null    object
```

```
df = train_df.copy()
```

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df.drop(columns=['Cabin'], inplace=True)
```

```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

```
/tmp/ipython-input-1457986264.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]
Ticket      0

df['Age'].fillna(df['Age'].median(), inplace=True)
/tmp/ipython-input-1457986264.py:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]
Embarked    2
dtype: int64
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

```
df['Fare'].fillna(df['Fare'].median(), inplace=True)
```

```
/tmp/ipython-input-146823877.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]

df['Fare'].fillna(df['Fare'].median(), inplace=True)
```

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(columns=['Survived', 'Name', 'Ticket', 'PassengerId'])
y = df['Survived']
```

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

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
      'Parch', 'Ticket', 'Fare', 'Embarked_Q', 'Embarked_S'],
      dtype='object')
```

```
param_grid_lr = {
'C': [0.01, 0.1, 1, 10],
'penalty': ['l2'],
'solver': ['liblinear']
}
```

```

}

lr = LogisticRegression(max_iter=1000)

grid_lr = GridSearchCV(
    lr,
    param_grid_lr,
    cv=5,
    scoring='accuracy'
)

grid_lr.fit(X_train, y_train)

best_lr = grid_lr.best_estimator_

y_pred_lr = best_lr.predict(X_test)

accuracy_score(y_test, y_pred_lr)

```

0.7877094972067039

```
X.isnull().sum()
```

	0
Pclass	0
Sex	0
Age	0
SibSp	0
Parch	0
Fare	0
Embarked_Q	0
Embarked_S	0

dtype: int64

```

param_grid_rf = {
    'n_estimators': [100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5]
}

rf = RandomForestClassifier(random_state=42)

grid_rf = GridSearchCV(
    rf,
    param_grid_rf,
    cv=5,
    scoring='accuracy'
)

grid_rf.fit(X_train, y_train)

best_rf = grid_rf.best_estimator_

y_pred_rf = best_rf.predict(X_test)

accuracy_score(y_test, y_pred_rf)

```

0.8156424581005587

```

param_grid_svm = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

svm = SVC()

grid_svm = GridSearchCV(
    svm,
    param_grid_svm,
    cv=5,
    scoring='accuracy'
)

grid_svm.fit(X_train, y_train)

best_svm = grid_svm.best_estimator_

y_pred_svm = best_svm.predict(X_test)

accuracy_score(y_test, y_pred_svm)

```

```
0.7821229050279329
```

```

from sklearn.feature_selection import SelectKBest, chi2

X_chi = X.copy()
X_chi[X_chi < 0] = 0

chi_selector = SelectKBest(score_func=chi2, k=5)
X_chi_selected = chi_selector.fit_transform(X_chi, y)

selected_chi_features = X.columns[chi_selector.get_support()]
selected_chi_features

```

```
Index(['Pclass', 'Sex', 'Age', 'Parch', 'Fare'], dtype='object')
```

```

from sklearn.feature_selection import RFE

rfe_model = LogisticRegression(max_iter=1000)

rfe = RFE(estimator=rfe_model, n_features_to_select=5)
X_rfe_selected = rfe.fit_transform(X, y)

selected_rfe_features = X.columns[rfe.support_]
selected_rfe_features

```

```
Index(['Pclass', 'Sex', 'SibSp', 'Embarked_Q', 'Embarked_S'], dtype='object')
```

```

X_rfe = df[selected_rfe_features]
y = df['Survived']

X_train_rfe, X_test_rfe, y_train_rfe, y_test_rfe = train_test_split(
    X_rfe, y, test_size=0.2, random_state=42
)

```

```

lr = LogisticRegression(max_iter=1000)

lr.fit(X_train_rfe, y_train_rfe)

y_pred_lr_rfe = lr.predict(X_test_rfe)

```

```
accuracy_score(y_test_rfe, y_pred_lr_rfe)
```

0.770949720670391

```
rf = RandomForestClassifier(random_state=42)

rf.fit(X_train_rfe, y_train_rfe)

y_pred_rf_rfe = rf.predict(X_test_rfe)

accuracy_score(y_test_rfe, y_pred_rf_rfe)
```

0.7932960893854749

```
svm = SVC()

svm.fit(X_train_rfe, y_train_rfe)

y_pred_svm_rfe = svm.predict(X_test_rfe)

accuracy_score(y_test_rfe, y_pred_svm_rfe)
```

0.7932960893854749

<https://www.kaggle.com/c/titanic/data>
fatimah aldighethair