In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('Titanic.csv')
df.head()
```

Out[2]:

	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
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [3]:
```
df.shape
```

Out[3]:
```
(891, 12)
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [5]:
```
NAs = pd.concat([df.isnull().sum()], axis=1, keys=["sum"])
NAs[NAs.sum(axis=1) > 0]
```

Out[5]:

	sum
Age	177
Cabin	687
Embarked	2

In [6]:
```
#Filling missing Age values with mean
df["Age"] = df["Age"].fillna(df["Age"].mean())
```

In [7]:
```
#Filling missing Embarked values with most common value
df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])
```

In [8]:
```
#Filling missing Cabin values with most common valus
df["Cabin"] = df["Cabin"].fillna(df["Cabin"].mode()[0])
```

In [9]:
```
df["Pclass"] = df["Pclass"].apply(str)
```

In [10]:
```
#Getting Dummies from all other categorical vars
for col in df.dtypes[df.dtypes == "object"].index:
    for_dummy = df.pop(col)
    df = pd.concat([df, pd.get_dummies(for_dummy, prefix = col)], axis = 1)
df.head()
```

Out[10]:

	PassengerId	Survived	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Name_Abbing, Mr. Anthony	...	Cabin_F G73	Cabin_F2	Cabin_F33	Cal
0	1	0	22.0	1	0	7.2500	0	0	1	0	...	0	0	0	
1	2	1	38.0	1	0	71.2833	1	0	0	0	...	0	0	0	
2	3	1	26.0	0	0	7.9250	0	0	1	0	...	0	0	0	
3	4	1	35.0	1	0	53.1000	1	0	0	0	...	0	0	0	
4	5	0	35.0	0	0	8.0500	0	0	1	0	...	0	0	0	

5 rows × 1733 columns

In [11]:
```
labels = df.pop("Survived")
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df, labels, test_size=0.33, random_state=42)
```

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [14]:
```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[14]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [15]:
```
y_pred = rf.predict(X_test)
```

In [16]:
```
 from sklearn.metrics import accuracy_score
```

In [17]:
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8101694915254237
```

In [18]:
```
from sklearn.model_selection import GridSearchCV
```

In [19]:
```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
```

In [20]:
```
# Create the GridSearchCV object
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')
```

In [21]:
```
grid_search.fit(X_train, y_train)
```

Out[21]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [22]:
```
# Get the best parameters and best estimator from the grid search
best_params = grid_search.best_params_
best_rf = grid_search.best_estimator_
```

In [23]:
```
# Make predictions on the test data using the best estimator
y_pred = best_rf.predict(X_test)
```

In [24]:
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Best Parameters:", best_params)
print("Best Accuracy:", accuracy)
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 300}
Best Accuracy: 0.8135593220338984
```

In [ ]:
```
```