Import the Libraries

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

Import dataset(csv file)

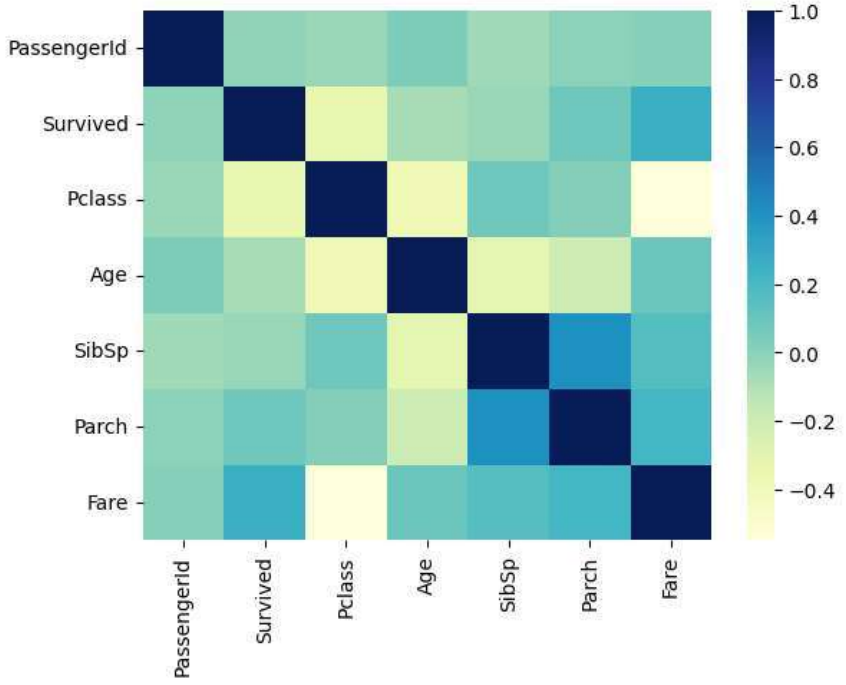
```
1 titanic_data = pd.read_csv("/content/train.csv")
2 titanic_data
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

Corrlation Heatmap

```
1 import seaborn as sns
2
3 sns.heatmap(titanic_data.corr(), cmap="YlGnBu")
4 plt.show()
```

```
<ipython-input-39-ea442d09ae1a>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future vers
  sns.heatmap(titanic_data.corr(), cmap="YlGnBu")
```



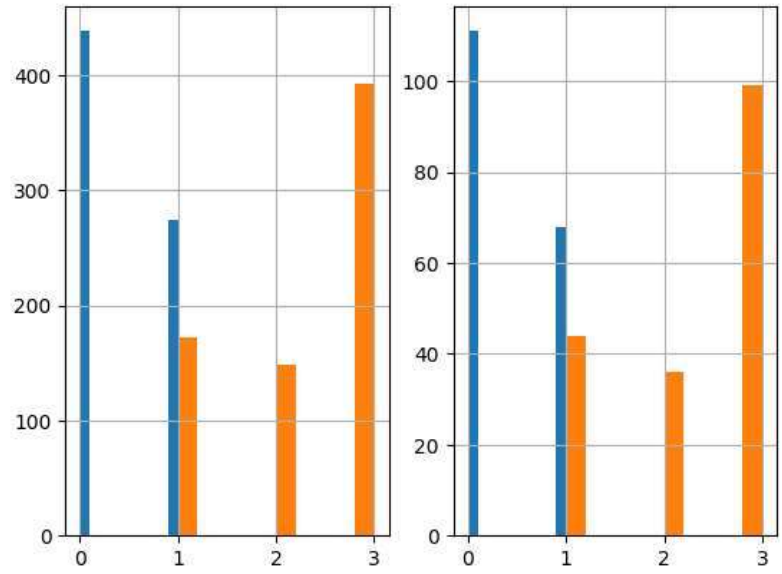
```
from sklearn.model_selection import StratifiedShuffleSplit

split=StratifiedShuffleSplit(n_splits=1,test_size=0.2)
for train_indices,test_indices in split.split(titanic_data,titanic_data[["Survived","Pclass","Sex"]]):
  strat_train_set=titanic_data.loc[train_indices]
  strat_test_set=titanic_data.loc[test_indices]
```

```
plt.subplot(1,2,1)
strat_train_set['Survived'].hist()
strat_train_set['Pclass'].hist()

plt.subplot(1,2,2)
strat_test_set['Survived'].hist()
strat_test_set['Pclass'].hist()

plt.show()
```




```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 871 to 459
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Name         712 non-null    object 
 4   Sex          712 non-null    object 
 5   Age          575 non-null    float64
 6   SibSp        712 non-null    int64  
 7   Parch        712 non-null    int64  
 8   Ticket       712 non-null    object 
 9   Fare         712 non-null    float64
 10  Cabin        161 non-null    object 
 11  Embarked     710 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 72.3+ KB
```

Estimators

```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer

class AgeImputer(BaseEstimator, TransformerMixin):

  def fit(self, X, y=None):
    return self

  def transform(self, X):
    imputer=SimpleImputer(strategy="mean")
    X['Age']=imputer.fit_transform(X[['Age']])
    return X
```

Encoding

```
from sklearn.preprocessing import OneHotEncoder

class FeatureEncoder(BaseEstimator, TransformerMixin):

  def fit(self, X, y=None):
    return self

  def transform(self, X):
    encoder=OneHotEncoder()
    matrix=encoder.fit_transform(X[['Embarked']]).toarray()

    column_names=["C","S","Q","N"]

    for i in range(len(matrix.T)):
      X[column_names[i]]=matrix.T[i]

    matrix = encoder.fit_transform(X[['Sex']]).toarray()

    column_names =  ["Female", "Male"]

    for i in range(len(matrix.T)):
      X[column_names[i]] = matrix.T[i]

    return X
```

Feature dropper

```
class FeatureDropper(BaseEstimator, TransformerMixin):

  def fit(self, X, y=None):
    return self

  def transform(self, X):
    return X.drop(["Embarked", "Name", "Ticket", "Cabin", "Sex","N"], axis=1, errors="ignore")

```

Pipeline

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([("ageimputer", AgeImputer()),
                     ("featureencoder", FeatureEncoder()),
                     ("featuredopper", FeatureDropper())])

```

```
strat_train_set = pipeline.fit_transform(strat_train_set)
```

```
strat_train_set
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
871	872	1	1	47.000000	1	1	52.5542	0.0	0.0	1.0	1.0	0.0
400	401	1	3	39.000000	0	0	7.9250	0.0	0.0	1.0	0.0	1.0
786	787	1	3	18.000000	0	0	7.4958	0.0	0.0	1.0	1.0	0.0
6	7	0	1	54.000000	0	0	51.8625	0.0	0.0	1.0	0.0	1.0
127	128	1	3	24.000000	0	0	7.1417	0.0	0.0	1.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
879	880	1	1	56.000000	0	1	83.1583	1.0	0.0	0.0	1.0	0.0
487	488	0	1	58.000000	0	0	29.7000	1.0	0.0	0.0	0.0	1.0
123	124	1	2	32.500000	0	0	13.0000	0.0	0.0	1.0	1.0	0.0
522	523	0	3	30.095513	0	0	7.2250	1.0	0.0	0.0	0.0	1.0
459	460	0	3	30.095513	0	0	7.7500	0.0	1.0	0.0	0.0	1.0

712 rows × 12 columns

```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 871 to 459
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Age          712 non-null    float64
 4   SibSp        712 non-null    int64  
 5   Parch        712 non-null    int64  
 6   Fare         712 non-null    float64
 7   C            712 non-null    float64
 8   S            712 non-null    float64
 9   Q            712 non-null    float64
 10  Female       712 non-null    float64
 11  Male         712 non-null    float64
dtypes: float64(7), int64(5)
memory usage: 72.3 KB
```

```
from sklearn.preprocessing import StandardScaler

X = strat_train_set.drop(['Survived'], axis=1)
y = strat_train_set['Survived']

scaler = StandardScaler()
X_data = scaler.fit_transform(X)
y_data = y.to_numpy()

```

Model selection

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

clf = RandomForestClassifier()

param_gird = [
    {"n_estimators" : [10, 100, 200, 500], "max_depth" : [None, 5, 10], "min_samples_split" : [2,3,4]}
]

grid_search = GridSearchCV(clf, param_gird, cv=3, scoring="accuracy", return_train_score=True)
grid_search.fit(X_data, y_data)

```

```
GridSearchCV
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [None, 5, 10],
                          'min_samples_split': [2, 3, 4],
                          'n_estimators': [10, 100, 200, 500]}],
             return_train_score=True, scoring='accuracy')
  estimator: RandomForestClassifier
  RandomForestClassifier()
    RandomForestClassifier
  RandomForestClassifier()
```

Result

```
1 final_clf = grid_search.best_estimator_
```

```
1 final_clf
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_split=4)
```

```
1 strat_test_set = pipeline.fit_transform(strat_test_set)
```

```
1 X_test = strat_test_set.drop(['Survived'], axis=1)
2 y_test = strat_test_set['Survived']
3
4 scaler = StandardScaler()
5 X_data_test = scaler.fit_transform(X_test)
6 y_data_test = y_test.to_numpy()
```

Model Accuracy

```
1 final_clf.score(X_data_test, y_data_test)
```

```
0.8324022346368715
```

```
1 final_data = pipeline.fit_transform(titanic_data)
```

```
1 final_data
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female
0	1	0	3	22.000000	1	0	7.2500	0.0	0.0	1.0	0.0
1	2	1	1	38.000000	1	0	71.2833	1.0	0.0	0.0	1.0
2	3	1	3	26.000000	0	0	7.9250	0.0	0.0	1.0	1.0
3	4	1	1	35.000000	1	0	53.1000	0.0	0.0	1.0	1.0
4	5	0	3	35.000000	0	0	8.0500	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	27.000000	0	0	13.0000	0.0	0.0	1.0	0.0
887	888	1	1	19.000000	0	0	30.0000	0.0	0.0	1.0	1.0
888	889	0	3	29.699118	1	2	23.4500	0.0	0.0	1.0	1.0
889	890	1	1	26.000000	0	0	30.0000	1.0	0.0	0.0	0.0
890	891	0	3	32.000000	0	0	7.7500	0.0	1.0	0.0	0.0

891 rows × 12 columns

```
X_final = final_data.drop(['Survived'], axis=1)
y_final = final_data['Survived']
```

```
prod_clf = RandomForestClassifier()

param_gird = [
    {"n_estimators" : [10, 100, 200, 500], "max_depth" : [None, 5, 10], "min_samples_split" : [2,3,4]}
]

grid_search = GridSearchCV(prod_clf, param_gird, cv=3, scoring="accuracy", return_train_score=True)
grid_search.fit(X_data_final, y_data_final)

```

```
GridSearchCV
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [None, 5, 10],
                          'min_samples_split': [2, 3, 4],
                          'n_estimators': [10, 100, 200, 500]}],
             return_train_score=True, scoring='accuracy')
  estimator: RandomForestClassifier
  RandomForestClassifier()
    RandomForestClassifier
    RandomForestClassifier()
```

Test data

```
prod_final_clf = grid_search.best_estimator_
```

Import Test dataset(csv file)

```
titanic_test_data = pd.read_csv("/content/test.csv")
```

```
final_test_data = pipeline.fit_transform(titanic_test_data)
```

```
X_final_test = final_test_data
X_final_test = X_final_test.fillna(method="ffill")

scaler = StandardScaler()
X_data_final_test = scaler.fit_transform(X_final_test)
```

```
predictions = prod_final_clf.predict(X_data_final_test)
```

```
final_df = pd.DataFrame(titanic_test_data['PassengerId'])
final_df['Survived'] = predictions
```

```
final_df
```

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0