Working with Titanic data set in Kaggle

The Challenge

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

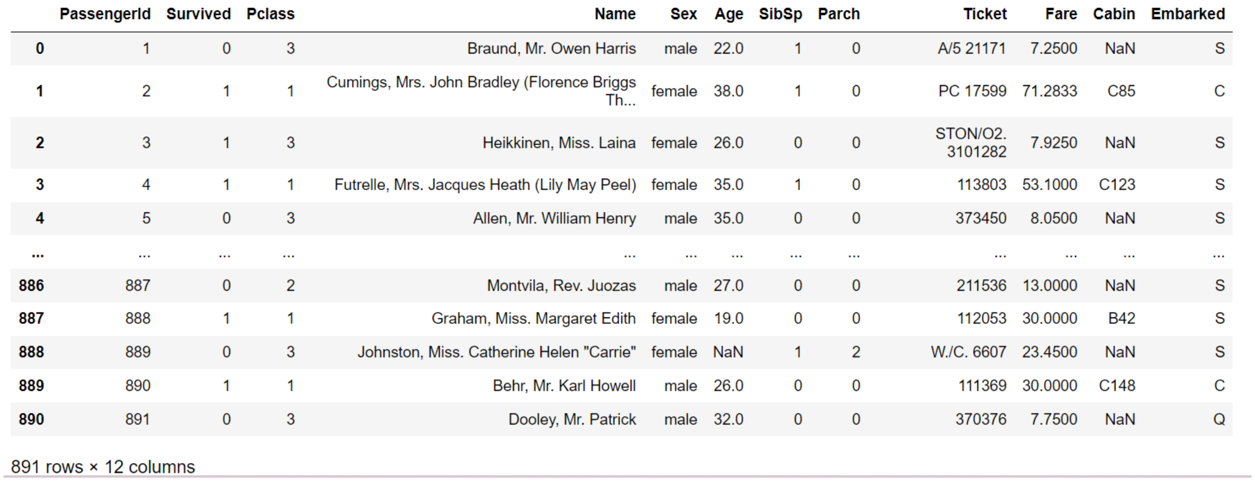
While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc.).

Analysis-

- Looking at the given variables (name ,age , seat no) we can see that some of these variables have no connection with the survival of a person example a name has nothing to do with someone’s survival

```
titanic_data=pd.read_csv("C:\\Users\\blaze\\Downloads\\titanic\\train.csv")
```



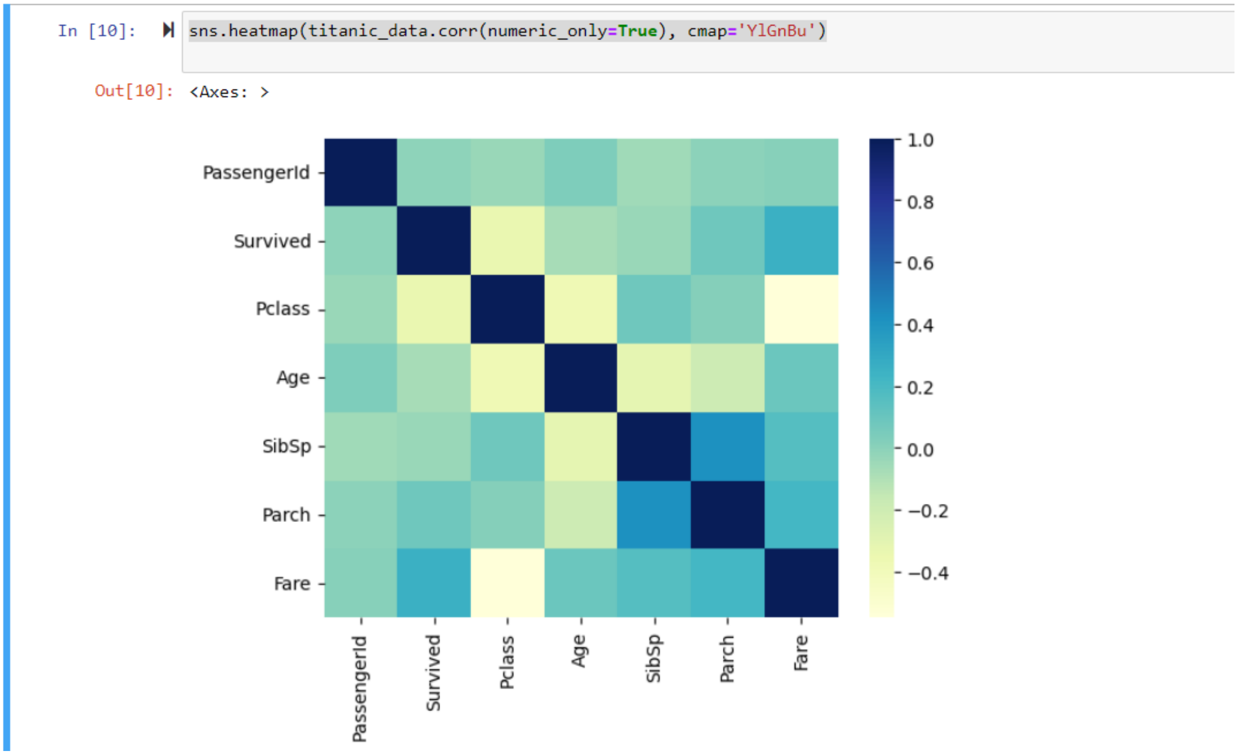
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

-

Correlation Heat map

```
import seaborn as sns
sns.heatmap(titanic_data.corr(numeric_only=True),cmap="YlGnBu")
```



```
In [10]: sns.heatmap(titanic_data.corr(numeric_only=True), cmap='YlGnBu')

Out[10]: <Axes: >
```

- The correlation heat map shows us the relation between variables (positively corelated or negatively)
- Here we are interested in the feature Survival ,hence we see correlation of other variables wrt survival
- We can see that fare is highly (positive) co related with survival
- Sometimes your testing data and training data can be little skewed so you want to make sure that the important features (fares) wrt to our prediction (survival) are equally distributed in test and training data set

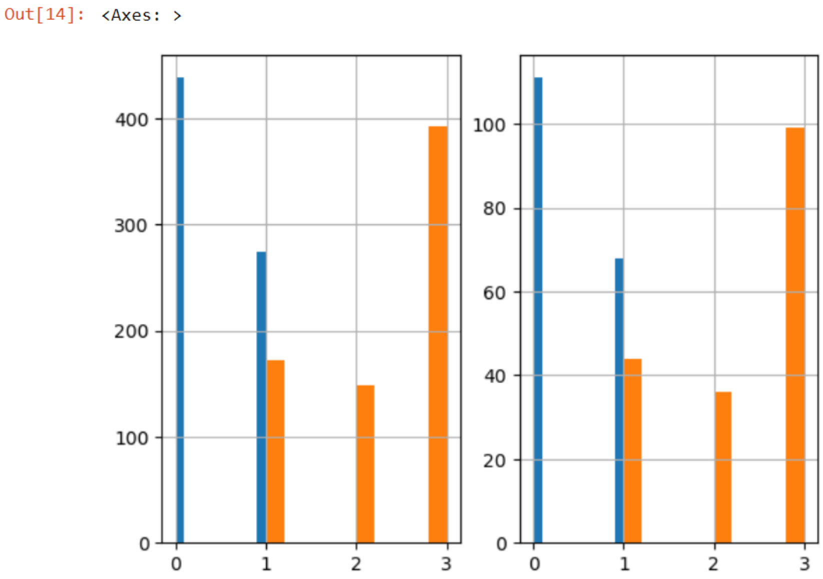
Stratified shuffle split

- For equal distribution of important features in testing and training data set we perform stratified shuffle split

```
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()
```

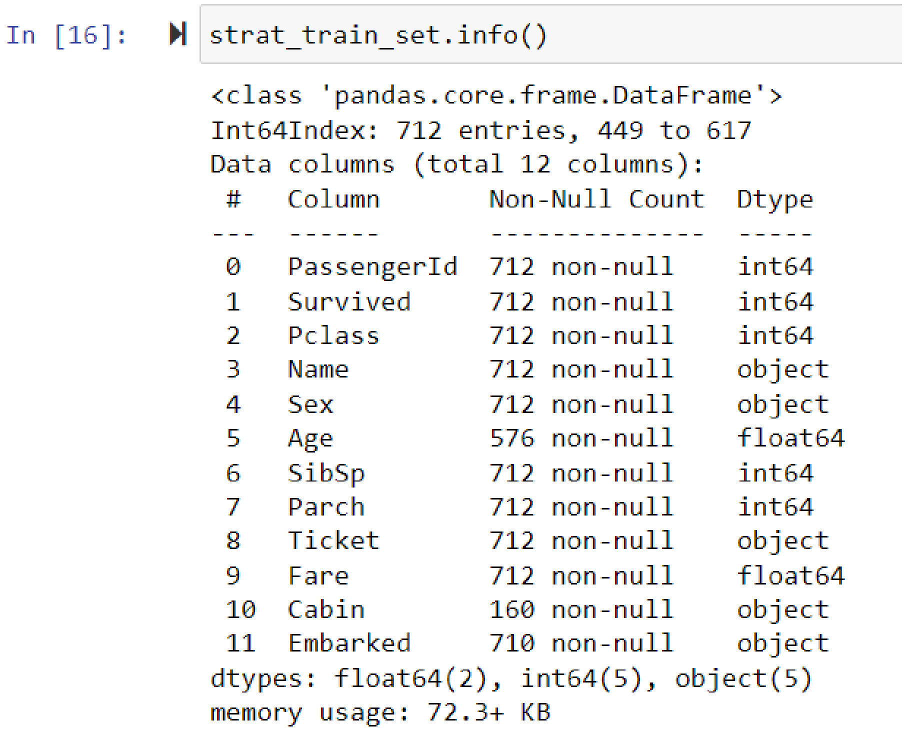
- The plots are made to clearly Visualize the equal distribution among the data set



```
Out[14]: <Axes: >
```

Imputation

- Imputation refers to the process of filling in missing values in a data set through various techniques
- How do you know/ check your data set has missing values



```
In [16]:    strat_train_set.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 449 to 617
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          576 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        160 non-null    object 
 11  Embarked     710 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 72.3+ KB
```

you can see age has only 576 values and other have high number so we can infer that it has missing values

```
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
```

- Here we use mean to fill missing values of age

One Hot -Encoding

- encoding refers to the process of transforming categorical variables into a numerical representation that can be used as input for machine learning algorithms. Most machine learning models require numerical input, so encoding categorical variables is necessary to incorporate them into the learning process.
- Here we convert feature (Sex) which is a string into a numerical value

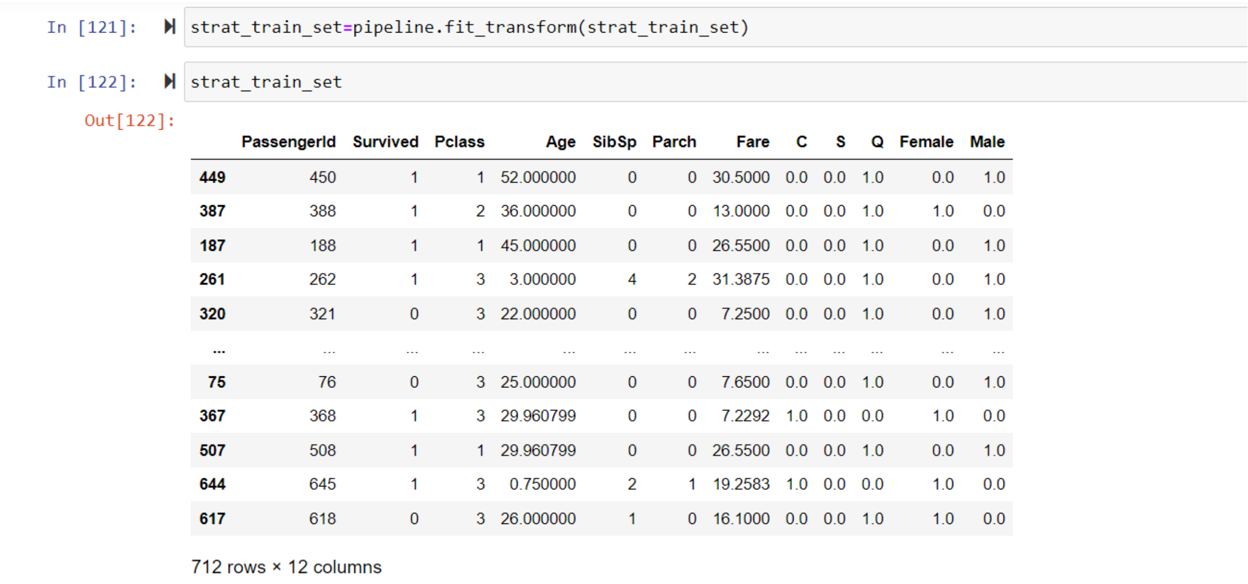
```
from sklearn.preprocessing import OneHotEncoder

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
```

Dropping Features

```
class FeatureDropper(BaseEstimator,TransformerMixin):
    def fir(self, X ,y=None):
        return self

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

Creating a Pipeline

```
from sklearn.pipeline import Pipeline

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



```
In [121]:  strat_train_set=pipeline.fit_transform(strat_train_set)
```

```
In [122]:  strat_train_set
```

Out[122]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
449	450	1	1	52.000000	0	0	30.5000	0.0	0.0	1.0	0.0	1.0
387	388	1	2	36.000000	0	0	13.0000	0.0	0.0	1.0	1.0	0.0
187	188	1	1	45.000000	0	0	26.5500	0.0	0.0	1.0	0.0	1.0
261	262	1	3	3.000000	4	2	31.3875	0.0	0.0	1.0	0.0	1.0
320	321	0	3	22.000000	0	0	7.2500	0.0	0.0	1.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
75	76	0	3	25.000000	0	0	7.6500	0.0	0.0	1.0	0.0	1.0
367	368	1	3	29.960799	0	0	7.2292	1.0	0.0	0.0	1.0	0.0
507	508	1	1	29.960799	0	0	26.5500	0.0	0.0	1.0	0.0	1.0
644	645	1	3	0.750000	2	1	19.2583	1.0	0.0	0.0	1.0	0.0
617	618	0	3	26.000000	1	0	16.1000	0.0	0.0	1.0	1.0	0.0

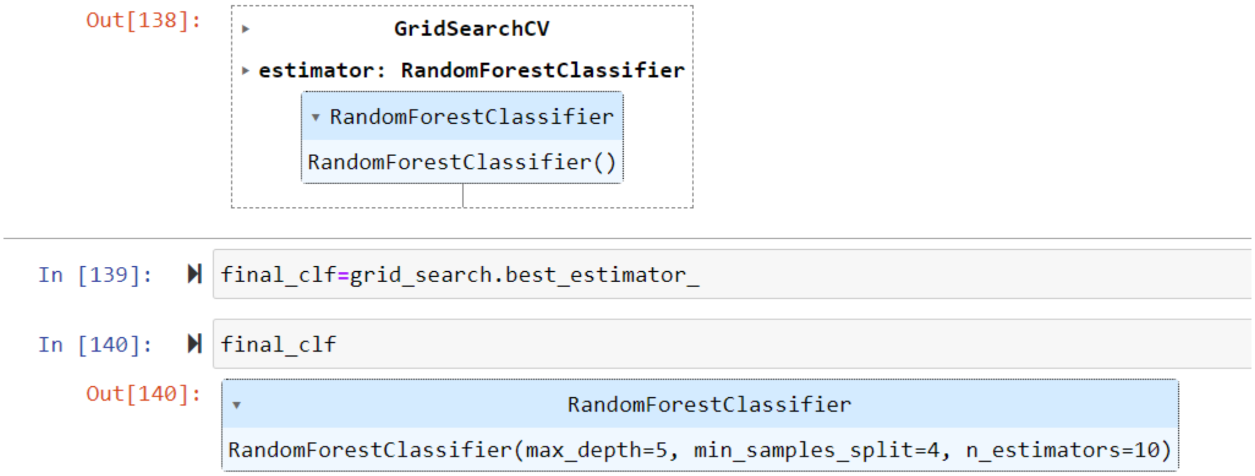
712 rows × 12 columns

Scaling

```
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()
```

Modal (Random Forest)

```
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)
```



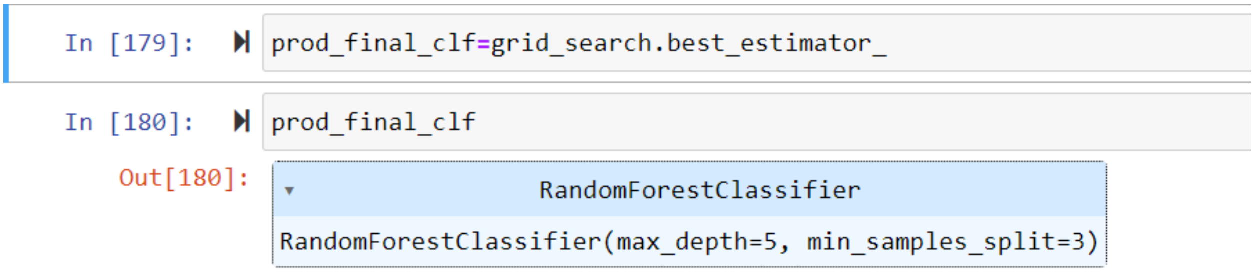
```
Out[138]:
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▾ RandomForestClassifier
  RandomForestClassifier()

In [139]: final_clf=grid_search.best_estimator_

In [140]: final_clf

Out[140]:
▾ RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_split=4, n_estimators=10)
```

- GridSearchCV is a tool used for optimizing hyperparameters here its takes all possible combination of parameters (n_estimators , max_depth , min samples) and gives us the best values of hyperparameters which can be used to train our model

Now we train our model on the entire training data -

```
final_data = pipeline.fit_transform(titanic_data)

X_final = final_data.drop (['Survived'], axis=1)
Y_final= final_data['Survived']
scaler = StandardScaler ()
X_data_final = scaler. fit_transform (X)
Y_data_final = y.to_numpy()
```

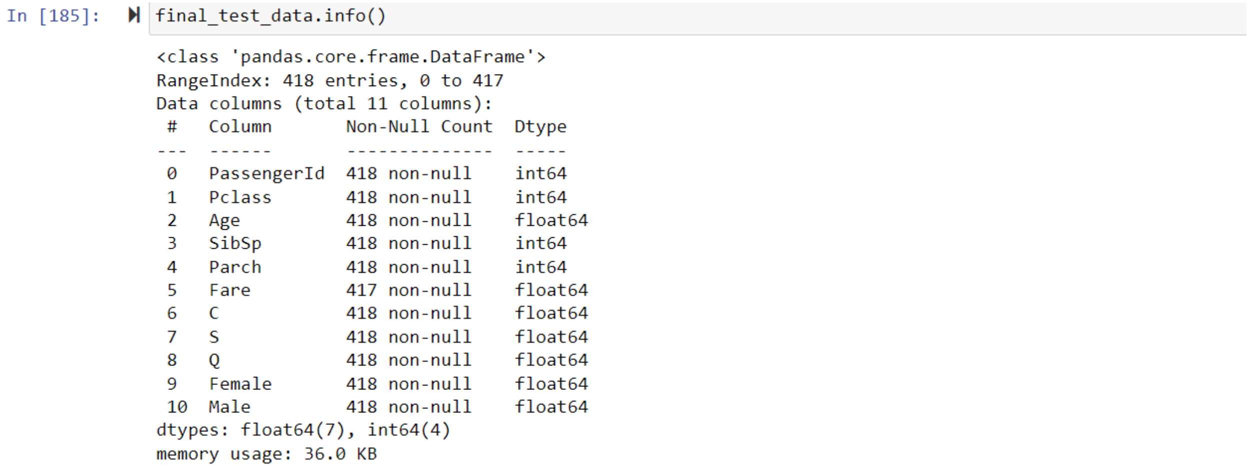
```
prod_clf = RandomForestClassifier ()

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)
```



```
In [179]:  prod_final_clf=grid_search.best_estimator_

In [180]:  prod_final_clf

Out[180]:  ▾ RandomForestClassifier
           RandomForestClassifier(max_depth=5, min_samples_split=3)
```

Working with Test Data

```
titanic_test_data=pd.read_csv("C:\\Users\\blaze\\Downloads\\titanic\\test.csv")

final_test_data=pipeline.fit_transform(titanic_test_data)
```

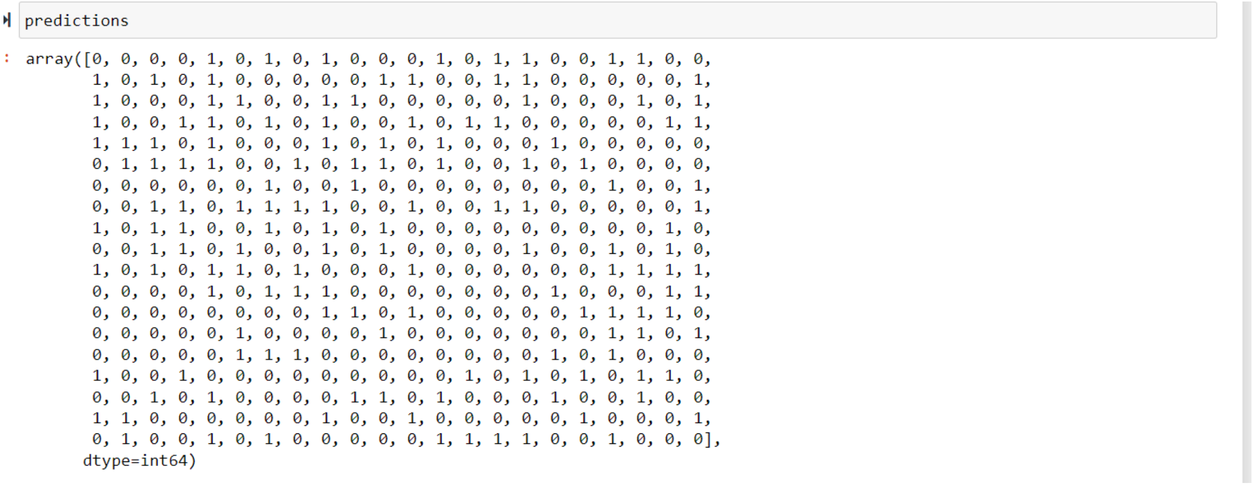


```
In [185]:  final_test_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Age          418 non-null    float64
 3   SibSp        418 non-null    int64  
 4   Parch        418 non-null    int64  
 5   Fare         417 non-null    float64
 6   C            418 non-null    float64
 7   S            418 non-null    float64
 8   Q            418 non-null    float64
 9   Female       418 non-null    float64
 10  Male         418 non-null    float64
dtypes: float64(7), int64(4)
memory usage: 36.0 KB
```

- We can see there is one null value in Fare

```
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)
```



```
predictions
```

```
: array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
         1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
         1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
         1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
         1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
         0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
         0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
         1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
         0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
         1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
         0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
         0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
         0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
         1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
         0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
         1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
         0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0],
        dtype=int64)
```

Creating a data frame for csv

```
final_df  = pd.DataFrame(titanic_test_data['PassengerId'])
final_df['Survived']=predictions
final_df.to_csv("C:\\Users\\blaze\\Downloads\\pred")
```