TITANIC PREDICTION USING DATA MINING AND MACHINE LEARNING

Course CSCI-555

Professor Dr. Othman Soufan

Shiva Priya Sunnam

x2022ela@stfx.ca

202206159

St. Francis Xavier University

1. Introduction to the Problem

The Titanic Problem comes from the event when the ship Titanic had sunk in the year 1912, causing many deaths. It is believed that approximately 1500 people were reported dead. As computer science students, we can look at this disaster as a great problem to be solved using Machine Learning Algorithms. Without any algorithmic knowledge, one can think of the various factors that might have led to the deaths. The first thing that comes to mind is that there weren't enough lifeboats or rescue resources to save the lives of all the people on the ship, knowing this fact it is obvious that only certain people were saved. We can think that factors like the age of a person, social class of the person or the importance of the person will affect the chances of his survival. Let's go further and understand the problem!

2. Dataset Analysis – Understanding What You Are Working With

The first and the most important step of solving any Data Mining and Machine Learning problem is evaluating and understanding your raw dataset.

The dataset provided by Kaggle for the Titanic Problem had the following specifications –

- **Dataset file type**: CSV
- **Training Dataset size**: 892 Rows, 12 Columns
- **Testing Dataset size**: 419 Rows, 11 Columns
- **Rows / Initial Features**:
    - PassengerId – Serial number of the Passenger Rows
    - Pclass – Passenger Class, can be considered as the Social Economic Class of the Passenger
    - Name – Name of the Passenger
    - Sex – Gender of the Passenger, contains two values.
    - Age – This is a float value.
    - SibSp – Indicates if the passenger has any siblings.
    - Parch – This information indicates the number of children.
    - Ticket – Gives the Ticket Number
    - Fare – Price of the Ticket
    - Cabin –Indicates if the Passenger has a Cabin or not.
    - Embarked – The port in which the Passenger boarded the Ship
      Only for Training Dataset:
    - Survived – Values are binary, indicating whether the Passenger survived.
- **Output to be Predicted:** 1 for Survived, 0 otherwise.
- On observing the values, we can see that many cells have empty values indicated with 'NA'. These values must be cleaned in the next step.
- **Is it a Binary Classification Problem?**
  As soon as we look at the output labels of the data, we can conclude that it is a binary classification problem. We now have an idea about the type of algorithms to be used.

3. Dataset Mining – Identifying What's Important

Many researchers and data scientists often say that a model is as good as the data it is being trained on. Machine learning algorithms learn from the data and the quality of the data that is being fed to the

algorithm will help build accuracy and precision. The data should be well-represented and must cover a wide range of cases. This will help the model learn what to do in each scenario. Data Mining helps us to discover patterns and behaviour of the data toward predicting the output. Doing this will help the model make better decisions toward prediction.

- Cleaning the Empty / NA Values

  On observing the dataset, we see that some cells are empty. Reading empty cells will throw errors in Python. The dataset will be read into a Pandas data frame and must not contain any empty values during evaluation.

- `cleanFeatureData(X)` Function:

```
def cleanFeatureData(X):
    X["Age"] = X["Age"].fillna(X["Age"].value_counts().idxmax())
    X["Sex"] = X["Sex"].replace({'male': 0, 'female': 1})
    X['Cabin'] = X["Cabin"].apply(lambda x: 0 if type(x) == float else 1)
    X["Embarked"] = X["Embarked"].fillna(X["Embarked"].value_counts().idxmax())
    return X
```

- This function fills the empty values in the Age column and replaces them with the most commonly found age in the column.
- It also cleans the Gender column into numerical format representing 0 for males and 1 for females.
- It checks if the person has a Cabin or not and converts that information into binary data.
- It fills all empty cells of the Embarked column to the most commonly occurring embarked port.

4. Feature Mining - Adding Features to Improve Predictions

The first set of experiments that I performed without feature engineering resulted in highly inaccurate results. This is because the features provided were not efficient to detect patterns. We perform Feature Mining was introduced.

Results before Feature Mining:

```
from sklearn.ensemble import RandomForestClassifier
p=[0.10, 0.20, 0.30, 0.40, 0.50]
for s in p:
  print()
  print('Split Proportion : ', s)
  X_train, X_test, y_train, y_test = splitDataset(s)
  s = s + 0.05
  model = RandomForestClassifier(n_estimators=50, max_depth=5, random_state=1, min_samples_leaf=8, min_samples_split=5)
  model.fit(X_train, y_train)
  y_pred=model.predict(X_test)
  print('Accuracy Score   : %.2f' % metrics.accuracy_score(y_test,y_pred))
  print('F1 Score         : %.2f' % metrics.f1_score(y_test, y_pred))
```

```
Split Proportion :  0.1
Accuracy Score   : 0.83
F1 Score         : 0.76

Split Proportion :  0.2
Accuracy Score   : 0.79
F1 Score         : 0.67

Split Proportion :  0.3
Accuracy Score   : 0.81
F1 Score         : 0.73

Split Proportion :  0.4
Accuracy Score   : 0.85
F1 Score         : 0.79

Split Proportion :  0.5
Accuracy Score   : 0.78
F1 Score         : 0.65
```

Even though the accuracy is above 80 percent the F1 Score was low, which means that the Recall Score was low as well, leading to Test Dataset Predictions being extremely inaccurate. To fix this problem, I have introduced Feature Mining into the solution.

- The function `addMinedFeatures(X)` will improvise the existing features to create a more stronger and diverse training dataset. The function does the following –
  - **Family Size**: Calculates the Family Size of each passenger and assigns them to labels 0, 1, and 2 based on the varying range. This is done because we can identify that passengers' family sizes can be simplified to three classes.
  - **Title**: Extracts the titles of the passengers and assigns a class for the titles. This is done because the titles also indicate the importance of the person in society. We can also say that the titles also tell us a passenger's social status.
  - **Single**: This column indicates if the passenger is single or with a family. We can think of this as an indicator because usually, one passenger with family might most probably try to rescue his family even when he has a chance to survive.
  - **FareRange**: This column takes the price of a ticket and assigns a class to it based on the range of the prices.
  - **AgeRange**: A class is assigned to the ages based on the division of the range.
  - **Embarked**: There are three ports of embarking, the function converts this data into the numerical format by assigning it to three classes.
  - **AgeAndFare**: This feature is simply the product of age and fare. This is to evaluate how age and fare taken together might affect the rates of survival.

Snapshot of the final Dataset after cleaning and feature mining:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Sex	SibSp	Parch	Cabin	Embarked	FamilySize	Title	Single	FareRange	AgeRange	FareAndAge
2	3	0	0	0	0	2	0	1	0	0	2	270.1074
3	3	1	1	0	0	0	0	3	0	0	3	329
4	2	0	0	0	0	2	0	1	0	0	4	600.625
5	3	0	0	0	0	0	0	1	0	0	2	233.8875
6	3	1	1	1	0	0	1	3	1	1	1	270.325
7	3	0	0	0	0	0	0	1	0	0	0	129.15
8	3	1	0	0	0	2	0	2	0	0	2	228.876

5. Experiments – The Real Fun Begins

I have used 5 different learning models to predict the outputs. Two algorithms are Tree based algorithms, the Random Forest Classifier and the Decision Tree Classier. One clustering algorithm, which predicts two clusters or two classes, which was the K Nearest Neighbors Algorithm. The other two algorithms are taken from Classification Algorithms, which are the Stochastic Gradient Descent and the Gaussian Process Classifier Algorithm.

- **Getting the Best Model Parameters**

At first, I manually used the brute force method to get the best parameters on which the learning models produce accurate results.

A Snapshot of Manually Testing with different Numbers of Estimators

```
Estimators       :  50      Estimators       :  550
Accuracy Score   : 0.82     Accuracy Score   : 0.80
F1 Score         : 0.75     F1 Score         : 0.72

Estimators       :  100     Estimators       :  600
Accuracy Score   : 0.81     Accuracy Score   : 0.80
F1 Score         : 0.73     F1 Score         : 0.72

Estimators       :  150     Estimators       :  650
Accuracy Score   : 0.82     Accuracy Score   : 0.80
F1 Score         : 0.74     F1 Score         : 0.72

Estimators       :  200     Estimators       :  700
Accuracy Score   : 0.81     Accuracy Score   : 0.80
F1 Score         : 0.73     F1 Score         : 0.72

Estimators       :  250     Estimators       :  750
Accuracy Score   : 0.82     Accuracy Score   : 0.80
F1 Score         : 0.74     F1 Score         : 0.72
```

- **Randomized Search CV**

I then discovered a package in sklearn called Randomized Search CV that gives the best parameters for each classifier with much less time complexity.

Snapshot of Running RandomizedSearchCV on Classifier Models

```
Best Parameters for  SGDClassifier are
 {'alpha': 0.1, 'l1_ratio': 0.5, 'learning_rate': 'optimal', 'loss': 'perceptron', 'max_iter': 1769, 'penalty': 'l1'}
Best Parameters for  GaussianProcessClassifier are
 {'warm_start': False, 'optimizer': 'fmin_l_bfgs_b', 'n_restarts_optimizer': 10, 'max_iter_predict': 300}

 Best Parameters for  DecisionTreeClassifier are
  {'criterion': 'gini', 'max_depth': 52, 'max_features': 0.7000000000000001, 'min_samples_leaf': 21
 Best Parameters for  KNeighborsClassifier are
  {'algorithm': 'brute', 'leaf_size': 73, 'n_neighbors': 9, 'p': 3, 'weights': 'uniform'}
```

When given a set of parameters, this model runs the classifier with all the combinations and then compares the performance of each one of them. It picks out the best parameters which give the best performance that can be used for our case.

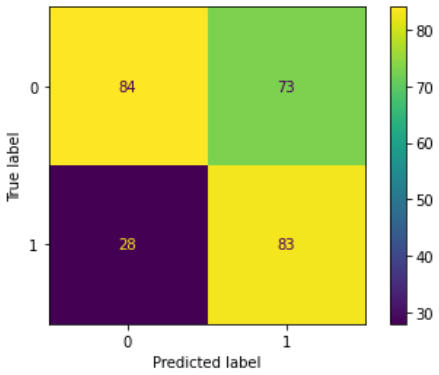
6. Picking a Classifier – How I got my Rank

After fine-tuning all the parameters, I passed the best parameters to the models and compared their performance. Below are comparisons of various algorithms –

- **Stochastic Gradient Descent Classifier**

```
SGDClassifier

Accuracy Score   : 0.62
F1 Score         : 0.62
Precision Score  : 0.53
Recall Score     : 0.75
ROC AUC Score    : 0.64
```



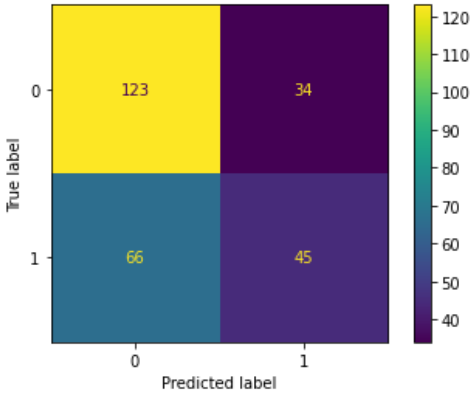
Parameters to Achieve this Score:

```
{'alpha': 0.1, 'l1_ratio': 0.5, 'learning_rate': 'optimal', 'loss':
'perceptron', 'max_iter': 1769, 'penalty': 'l1'}
```

- **Gaussian Process Classifier**

```
GaussianProcessClassifier

Accuracy Score   : 0.63
F1 Score         : 0.47
Precision Score  : 0.57
Recall Score     : 0.41
ROC AUC Score    : 0.59
```



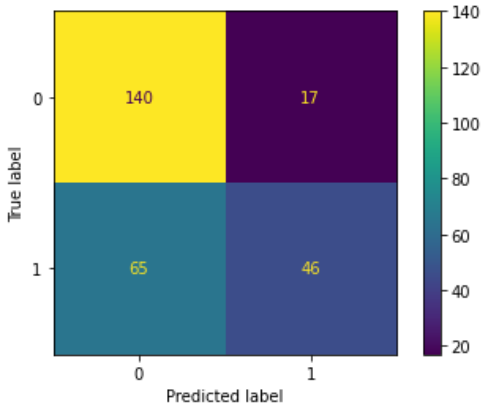
Parameters to Achieve this Score:

```
{'warm_start':       False,       'optimizer':       'fmin_l_bfgs_b',
'n_restarts_optimizer': 10, 'max_iter_predict': 300}
```

- **K Nearest Neighbors Classifier**

```
KNeighborsClassifier

Accuracy Score   : 0.69
F1 Score         : 0.53
Precision Score  : 0.73
Recall Score     : 0.41
ROC AUC Score    : 0.65
```



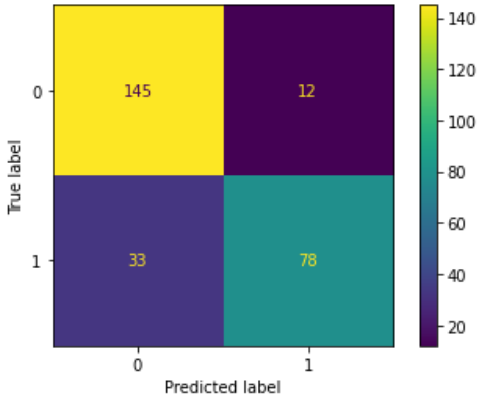
Parameters to Achieve this Score:

```
{'algorithm': 'brute', 'leaf_size': 73, 'n_neighbors': 9, 'p': 3,
'weights': 'uniform'}
```

- **Random Forest Classifier**

```
RandomForestClassifier

Accuracy Score   : 0.83
F1 Score         : 0.78
Precision Score  : 0.87
Recall Score     : 0.70
ROC AUC Score    : 0.81
```



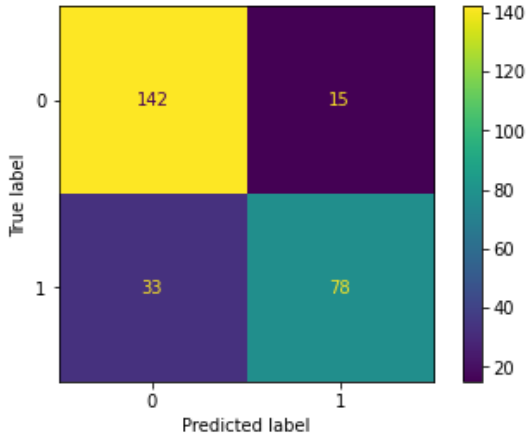
Parameters to Achieve this Score:

```
{'max_depth': 31, 'max_features': 0.6, 'min_samples_leaf': 2,
'min_samples_split': 65, 'n_estimators': 965, 'random_state': 281}
```

- **Decision Tree Classifier**

```
DecisionTreeClassifier

Accuracy Score   : 0.82
F1 Score         : 0.76
Precision Score  : 0.84
Recall Score     : 0.70
ROC AUC Score    : 0.80
```



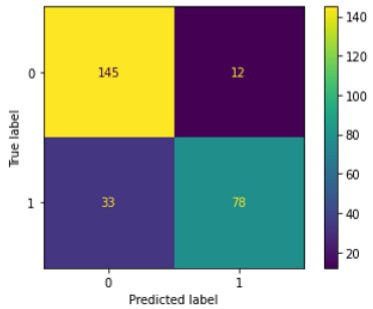
Parameters to Achieve this Score:

```
{'criterion': 'gini', 'max_depth': 52, 'max_features': 0.7000000000000001, 'min_samples_leaf': 21, 'min_samples_split': 74, 'random_state': 171}
```

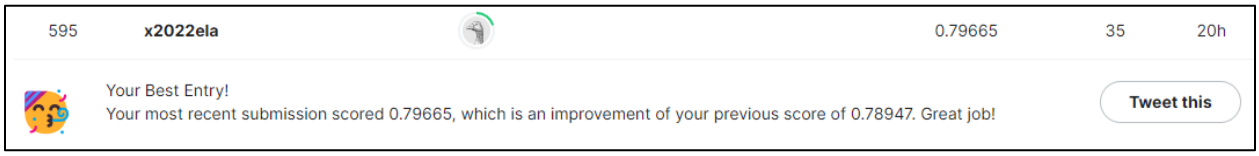
7. Final Results – Numbers, Numbers & Numbers!

After comparing the Metrics of all 5 models, we can see that the Random Forest Classifier has performed best. With an ***Accuracy of 83, F1 Score of 78, Precision of 87, Recall of 70 and ROC AUC Score of 81***.

Confusion Matrix for Random Forest:



The Kaggle Accuracy of this score was ***79.665%***.



8. Reproducibility

- Run Manually on local

For reproducing the models and results, download the ***titanic.zip*** file and extract it. Now enter the command prompt/terminal of the folder and run the following command:

```
python .\titanicPredictionModels.py
```

- Run With DOCKER:

  Simply run the 'run.sh' file using the command ***sh run.sh,*** this will build and run the Dockerfile and Docker Container

The results should look like this:

```
PS C:\Users\shiva\Documents\DMML\titanic\titanic> python .\titanicPredictionModels.py
Best Parameters for  RandomForestClassifier are
 {'max_depth': 31, 'max_features': 0.6, 'min_samples_leaf': 2, 'min_samples_split': 65, 'n_estimators': 965, 'random_state': 281}
Best Parameters for  DecisionTreeClassifier are
 {'criterion': 'gini', 'max_depth': 52, 'max_features': 0.7000000000000001, 'min_samples_leaf': 21, 'min_samples_split': 74, 'random_state': 171}
Best Parameters for  KNeighborsClassifier are
 {'algorithm': 'brute', 'leaf_size': 73, 'n_neighbors': 9, 'p': 3, 'weights': 'uniform'}
Best Parameters for  SGDClassifier are
 {'warm_start': False, 'optimizer': 'fmin_l_bfgs_b', 'n_restarts_optimizer': 10, 'max_iter_predict': 300}

KNeighborsClassifier

Accuracy Score   : 0.70
F1 Score         : 0.54
Precision Score  : 0.73
Recall Score     : 0.42
ROC AUC Score    : 0.66

DecisionTreeClassifier

Accuracy Score   : 0.82
F1 Score         : 0.76
Precision Score  : 0.84
Recall Score     : 0.70
ROC AUC Score    : 0.80

SGDClassifier

Accuracy Score   : 0.63
F1 Score         : 0.61
Precision Score  : 0.54
Recall Score     : 0.68
ROC AUC Score    : 0.64

GaussianProcessClassifier

Accuracy Score   : 0.63
F1 Score         : 0.47
Precision Score  : 0.57
Recall Score     : 0.41
ROC AUC Score    : 0.59

Best Classifier Model & Metrics:

RandomForestClassifier

Accuracy Score   : 0.83
F1 Score         : 0.78
Precision Score  : 0.87
Recall Score     : 0.70
ROC AUC Score    : 0.81
PS C:\Users\shiva\Documents\DMML\titanic\titanic>
```

The code automatically downloads the output files as follows:

```
-a----        2023-01-27   9:47 PM          13637 FinalDataset.csv
-a----        2023-01-27   9:47 PM           3258 RandomForestClassifier.csv
```

GitHub for the Code: https://github.com/shivapriyasunnam/titanic-prediction-using-ml

9. References

i. Kaggle Discussion Forums - https://www.kaggle.com/competitions/titanic/discussion

ii. Sklearn Documentation - https://scikit-learn.org/stable/index.html