A

Mini Project Report

On

"Trip History Analysis"

Mr. Shinde Pranit Sanjay

Roll Number:4229

Mr .Raul Ketan Gopalsing

Roll Number:4262

B.E. Division B

Mini Project Guide

Prof. M. B. Vaidya

For the partial fulfilment of
Mini Project Work in B.E. Computer Engineering

Department of Computer Engineering
Amrutvahini College of Engineering
Sangamner

2021-22

Department of Computer Engineering

Amrutvahini College of Engineering, Sangamner

CERTIFICATE

This is to certify that **Mr. Shinde Pranit Sanjay** and **Mr. Raul Ketan Gopalsing** student in Final Year Division B, Computer Engineering has successfully completed Mini Project titled **" Trip History Analysis"** at Amrutvahini College of Engineering, Sangamner towards fulfillment of Mini Project Work in Computer Engineering.

Prof. M.B.Vaidya
Project Guide

Prof. M.A.Wakchaure
Project Coordinator

Prof. R. L. Paikrao
Head of Department

Dr. M. A. Venkatesha
Principal

Abstract

Trip History Analysis Use trip history dataset that is from a bike sharing service in the United States. The data is provided quarter-wise from 2010 (Q4) onwards. Each file has 7 columns. Predict the class of user.Make use of at least two classification algorithms and provide comparative analysis. The bike sharing companies basically gather the data of different types of users and use it to track which members are taking the rental bikes and to which location. By using this data they can decide the to increase thebikes for particular route,give some discounts to regular members. Using this data we are performing analysis to find out member type of users

Keywords - Machine Learning, Decision Tree,Logistic Regression,Gaussian Naive Bayes

Acknowledgements

Achievement is Finding out what you have been doing and what you have to do. The higher is submit,the harder is climb. The goal was fixed and I began with the determined resolved and put in a ceaseless sustained hard work. Greater the challenge, greater was our determination and it guided us to overcome all difficulties. It has been rightly said that we are built on the shoulders of others. For everything I have achieved, the credit goes to who had really help us to complete this seminar and for the timely guidance and infrastructure. Before we proceed any further, We would like to thank all those who have helped me in all the way through. To start with I thank my guide Prof.M.B.Vaidya, for his guidance, care and support, which he offered whenever I needed it the most. I would also like to take this opportunity to thank to Mini Project Coordinator Prof. M.A.Wakchaure and our respected Head of Department Prof. R. L. Paikrao. I also thankful to Honorable Principal Dr. M. A. Venkatesh sir for his encouragement and support.

Contents

List of Figures

Chapter 1

Introduction

1.In this project we are going to analyze the Capital Bike Share Dataset and perform analysis on the same.

2.When a rental occurs within the system software collects basic data about the trip. That data can be exported from our system and used for various types of analysis or research.

Objective : We have to perform analysis on trip history dataset and predict the type of member whether casual or member.

Chapter 2

System Architecture

2.1 Architecture

System Architecture :

1. System architecture basically includes the Jupyter Notebook, Python and the dataset.The System basically calculates the accuracy for different classification algorithm and compares the same.

2.Each .csv file contains data for one quarter of the year. Within each file there are 7 columns.

Duration - Duration of trip

Start date – Includes start date and time

End date – Includes end date and time

Start station – Includes starting station name and number

End station – Includes ending station name and number

Bike - Includes ID number of bike used for the trip

Member Type – Lists whether user was a Registered (annual or monthly) or Casual (1 to 5 day) member.

3.So using the above features and matrix we found out the features useful for predicting to member-type like station number,end station and duration.

4.Using the different algorithm we can find out the accuracy and compare it

Algorihms Used: 1.Decision Tree Classifier

2.Logistic Regression

3.Gaussian Naive Bay

2.2 Output and Source Code

In [2]:
```
import numpy as np
import pandas as pd
%matplotlib inline
from sklearn import tree, linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from  sklearn.metrics  import accuracy_score
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
```

In [3]:
```
data=pd.read_csv('tripdata.csv')
```

In [4]:
```
data.head()
```

Out[4]:

	Duration	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member type
0	552	2018-01-01 00:05:06	2018-01-01 00:14:18	31104	Adams Mill & Columbia Rd NW	31400	Georgia & New Hampshire Ave NW	W00886	Member
1	1282	2018-01-01 00:14:30	2018-01-01 00:35:53	31321	15th St & Constitution Ave NW	31321	15th St & Constitution Ave NW	W01435	Casual
2	1265	2018-01-01 00:14:53	2018-01-01 00:35:58	31321	15th St & Constitution Ave NW	31321	15th St & Constitution Ave NW	W21242	Casual
3	578	2018-01-01 00:15:31	2018-01-01 00:25:09	31406	14th & Upshur St NW	31103	16th & Harvard St NW	W21322	Casual
4	372	2018-01-01	2018-01-01	31618	4th & East Capitol St NE	31619	Lincoln Park / 13th & East Capitol	W00119	Member

FIGURE 2.1: Source Code

Out[4]:

	Duration	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member type
0	552	2018-01-01 00:05:06	2018-01-01 00:14:18	31104	Adams Mill & Columbia Rd NW	31400	Georgia & New Hampshire Ave NW	W00886	Member
1	1282	2018-01-01 00:14:30	2018-01-01 00:35:53	31321	15th St & Constitution Ave NW	31321	15th St & Constitution Ave NW	W01435	Casual
2	1265	2018-01-01 00:14:53	2018-01-01 00:35:58	31321	15th St & Constitution Ave NW	31321	15th St & Constitution Ave NW	W21242	Casual
3	578	2018-01-01 00:15:31	2018-01-01 00:25:09	31406	14th & Upshur St NW	31103	16th & Harvard St NW	W21322	Casual
4	372	2018-01-01 00:18:02	2018-01-01 00:24:15	31618	4th & East Capitol St NE	31619	Lincoln Park / 13th & East Capitol St NE	W00119	Member

In [5]:
```
data.dtypes
```

Out[5]:
```
Duration                int64
Start date             object
End date               object
Start station number    int64
Start station          object
End station number      int64
End station            object
Bike number            object
Member type            object
dtype: object
```

In [6]:
```
data = data.drop('Start date',axis=1)
data = data.drop('End date',axis=1)
data = data.drop('Start station',axis=1)
data = data.drop('End station',axis=1)
data = data.drop('Bike number',axis=1)
```

In [6]:
```
data = data.drop('Start date',axis=1)
data = data.drop('End date',axis=1)
data = data.drop('Start station',axis=1)
data = data.drop('End station',axis=1)
data = data.drop('Bike number',axis=1)
```

In [7]:
```
data.head()
```

Out[7]:

	Duration	Start station number	End station number	Member type
0	552	31104	31400	Member
1	1282	31321	31321	Casual
2	1265	31321	31321	Casual
3	578	31406	31103	Casual
4	372	31618	31619	Member

In [8]:
```
# convert Member type to int value representation
# 1 - Registered Memeber
# 0 - Casual Member

le = LabelEncoder()
le.fit(data['Member type'])
data['Member type'] = le.transform(data['Member type'])
```

In [9]:
```
data.head()
```

Out[9]:

	Duration	Start station number	End station number	Member type

In [9]:
```
data.head()
```

Out[9]:

	Duration	Start station number	End station number	Member type
0	552	31104	31400	1
1	1282	31321	31321	0
2	1265	31321	31321	0
3	578	31406	31103	0
4	372	31618	31619	1

In [10]:
```
data.shape
```

Out[10]:
```
(168590, 4)
```

In [11]:
```
X = data.drop(['Member type'],axis=1).values
y = data['Member type'].values
X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size=0.3)
```

In [63]:
```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
lr_predicted = lr.predict(X_test)
lr_acc = accuracy_score(y_test,lr_predicted)
print("Accuracy:",lr_acc)
```

```
Accuracy: 0.9108488047926924
```

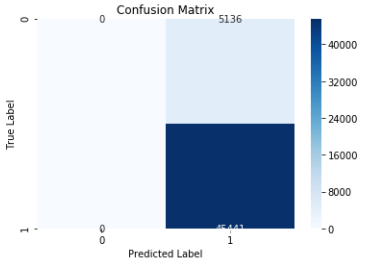
In [14]:
```
conf_matrix = confusion_matrix(y_test, lr_predicted)
sns.heatmap(conf_matrix, annot=True,fmt="d",cmap='Blues')
```

In [63]:
```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
lr_predicted = lr.predict(X_test)
lr_acc = accuracy_score(y_test,lr_predicted)
print("Accuracy:",lr_acc)
```

```
Accuracy: 0.9108488047926924
```

In [14]:
```
conf_matrix = confusion_matrix(y_test, lr_predicted)
sns.heatmap(conf_matrix, annot=True,fmt="d",cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[14]:
```
Text(0.5, 1, 'Confusion Matrix')
```



In [15]:

```
confusion_matrix(y_test, lr_predicted)

tn_lr = confusion_matrix(y_test, lr_predicted)[0,0]
fp_lr = confusion_matrix(y_test, lr_predicted)[0,1]
tp_lr = confusion_matrix(y_test, lr_predicted)[1,1]
fn_lr = confusion_matrix(y_test, lr_predicted)[1,0]

precison_lr = tp_lr/(tp_lr+fp_lr)
recall_lr = tp_lr/(tp_lr+fn_lr)

print("Precision: ", precison_lr)
print("Recall: ", recall_lr)
```

```
Precision:  0.8984518654724479
Recall:  1.0
```

In [61]:

```
GNB = GaussianNB()
GNB.fit(X_train, y_train)
GNB_predicted = GNB.predict(X_test)
GNB_acc = accuracy_score(y_test,GNB_predicted)
print("Accuracy:",GNB_acc)
```

```
Accuracy: 0.907942345334836
```

In [55]:

```
conf_matrix = confusion_matrix(y_test, GNB_predicted)
sns.heatmap(conf_matrix, annot=True,fmt="d",cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[55]:

```
Text(0.5, 1, 'Confusion Matrix')
```

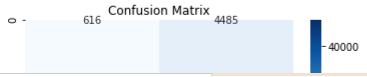


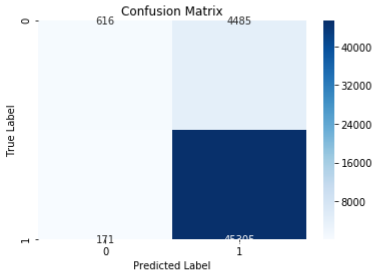
FIGURE 2.2: Source Code

In [55]:

```
conf_matrix = confusion_matrix(y_test, GNB_predicted)
sns.heatmap(conf_matrix, annot=True,fmt="d",cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[55]:

```
Text(0.5, 1, 'Confusion Matrix')
```



In [56]:

```
tn_GNB = confusion_matrix(y_test, GNB_predicted)[0,0]
fp_GNB = confusion_matrix(y_test, GNB_predicted)[0,1]
tp_GNB = confusion_matrix(y_test, GNB_predicted)[1,1]
fn_GNB = confusion_matrix(y_test, GNB_predicted)[1,0]

precison_GNB = tp_GNB/(tp_GNB+fp_GNB)
recall_GNB = tp_GNB/(tp_GNB+fn_GNB)

print("Precision: ", precison_GNB)
print("Recall: ", recall_GNB)
```

In [56]:
```
tn_GNB = confusion_matrix(y_test, GNB_predicted)[0,0]
fp_GNB = confusion_matrix(y_test, GNB_predicted)[0,1]
tp_GNB = confusion_matrix(y_test, GNB_predicted)[1,1]
fn_GNB = confusion_matrix(y_test, GNB_predicted)[1,0]

precison_GNB = tp_GNB/(tp_GNB+fp_GNB)
recall_GNB = tp_GNB/(tp_GNB+fn_GNB)

print("Precision: ", precison_GNB)
print("Recall: ", recall_GNB)
```

```
Precision:  0.9099216710182768
Recall:  0.996239774826282
```

In [60]:
```
DTC = tree.DecisionTreeClassifier(max_depth=10)
DTC.fit(X_train, y_train)
DTC_predicted = DTC.predict(X_test)
DTC_acc = accuracy_score(y_test,DTC_predicted)
print("Accuracy:",DTC_acc)
```

```
Accuracy: 0.9228503074520039
```

In [58]:
```
conf_matrix = confusion_matrix(y_test, DTC_predicted)
sns.heatmap(conf_matrix, annot=True,fmt="d",cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[58]:
```
Text(0.5, 1, 'Confusion Matrix')
```

In [59]:
```
tn_DTC = confusion_matrix(y_test, DTC_predicted)[0,0]
fp_DTC = confusion_matrix(y_test, DTC_predicted)[0,1]
tp_DTC = confusion_matrix(y_test, DTC_predicted)[1,1]
fn_DTC = confusion_matrix(y_test, DTC_predicted)[1,0]

precison_DTC = tp_DTC/(tp_DTC+fp_DTC)
recall_DTC = tp_DTC/(tp_DTC+fn_DTC)

print("Precision: ", precison_DTC)
print("Recall: ", recall_DTC)
```

```
Precision:  0.9391333741786914
Recall:  0.9774826281994898
```

In [65]:
```
models = [('Logistic Regression', tp_lr, fp_lr, tn_lr, fn_lr, lr_acc),
          ('Gaussian Naive Bayes', tp_GNB, fp_GNB, tn_GNB, fn_GNB, GNB_acc),
          ('Decision Tree Classifier', tp_DTC, fp_DTC, tn_DTC, fn_DTC, DTC_acc)
         ]

predict = pd.DataFrame(data = models, columns=['Model', 'True Positive', 'False Positive', 'True Negative',
                                               'False Negative', 'Accuracy'])
predict
```

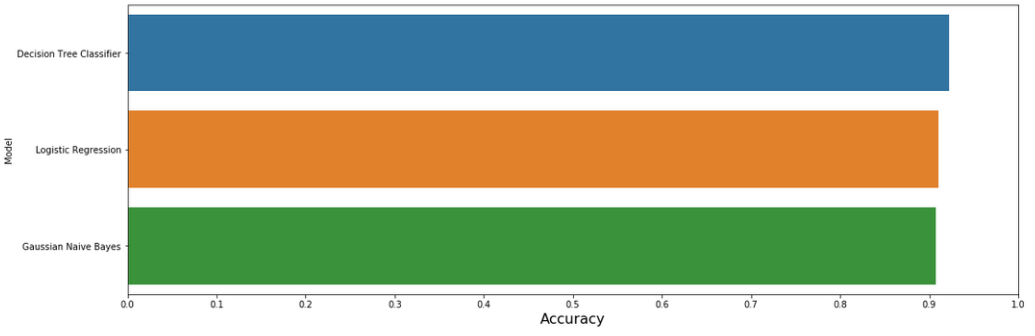
Out[65]:

	Model	True Positive	False Positive	True Negative	False Negative	Accuracy
0	Logistic Regression	45267	4300	801	209	0.910849
1	Gaussian Naive Bayes	45305	4485	616	171	0.907942
2	Decision Tree Classifier	44452	2881	2220	1024	0.922850

In [66]:
```
f, axe = plt.subplots(1,1, figsize=(18,6))
predict.sort_values(by=['Accuracy'], ascending=False, inplace=True)
sns.barplot(x='Accuracy', y='Model', data = predict, ax = axe)
```


1	Gaussian Naive Bayes	45305	4485	616	171	0.907942
2	Decision Tree Classifier	44452	2881	2220	1024	0.922850

In [66]:
```
f, axe = plt.subplots(1,1, figsize=(18,6))
predict.sort_values(by=['Accuracy'], ascending=False, inplace=True)
sns.barplot(x='Accuracy', y='Model', data = predict, ax = axe)
axe.set_xlabel('Accuracy', size=16)
axe.set_ylabel('Model')
axe.set_xlim(0,1.0)
axe.set_xticks(np.arange(0, 1.1, 0.1))
plt.show()
```



Chapter 3

Hardware and Software Requirements

3.1 Hardware Requirements

1. Min 500 GB HDD
2. Min 4GB RAM
3. Processor i3

3.2 Software Requirements

1. 32/64 bit Operating System
2. Python 3 and above
3. Jupyter Notebook
4. Anaconda

Chapter 4

Result

The Cross Validation Scores for Various models are:

Model	Cross-Validation Score
Logistic Regression	0.910849
Gaussian Naive Bayes	0.907942
Decision Tree Classifier	0.922850

TABLE 4.1: Cross Validation Scores for vaious Models

We see that Decision Tree Classifier gives the best score. We then use this model to perform training and testing of the model. After training, the model gives an accuracy of 92.20 %.

Chapter 5

Conclusion

We used different classification algorithms and the results obtained were
The accuracy score achieved using Logistic Regression is: 91.08
The accuracy score achieved using Gaussian Naive Bayes is: 90.79
The accuracy score achieved using Decision Tree Classifier is: 92.28
Based on the above results we can conclude that the accuracy is higher for Decision Tree Classifier classification algorithm and can further be increased by inclreaseing the dataset size and a bit of preprocessing of data.Also we have found out that the Member type is dependent on the Start Station,End Station and the Duration.

References

[1] **Dataset:** https://www.capitalbikeshare.com/system-data

[2] Ahas, R.; Aasa, A.; Mark, Ü.; Pae, T.; Kull, A. (2007). Seasonal tourism spaces in estonia: Case study with mobile positioning data. Tourism management, 28(3):898–910.

[3] Alessandretti, L.; Aslak, U.; Lehmann, S. (2020). The scales of human mobility. Nature, 587(7834):402–407.

[4] Alexander, L.; Jiang, S.; Murga, M.; González, M. C. (2015). Origin– destination trips by purpose and time of day inferred from mobile phone data.