

**PUNE INSTITUTE OF COMPUTER TECHNOLOGY  
DHANKAWADI, PUNE**

**DATA ANALYTICS MINI-PROJECT REPORT  
ON**

**“TRIP HISTORY ANALYSIS ”**

**SUBMITTED BY**

<b>Omkar Amilkanthawar</b>	<b>41403</b>
<b>Aniruddha Deshmukh</b>	<b>41405</b>
<b>Atharva Satpute</b>	<b>41409</b>

**Under the guidance of**  
Prof. Bhumesh Masram



**DEPARTMENT OF COMPUTER ENGINEERING  
Academic Year 2021-22**

---

# Contents

**1 Problem Statement1**

**2 Abstract2**

**3 Hardware and Software Requirements1**

3.1 Hardware Requirements ..... 1

3.2 Software Requirements..... 1

**4 INTRODUCTION2**

**5 OBJECTIVE3**

**6 Scope4**

**7 System Architecture5**

**8 Result6**

**9 Conclusion7**

**References8**

---

# 1 Problem Statement

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.

---

## 2 Abstract

Data Analytics is the process of analyzing data and drawing conclusion from it. One such dataset is the trip history of capital bikeshares, which logs the travel history of its riders. The dataset is available for each quarter after 2010. We select the first quarter of 2017 for our analysis. The main goal is predict the class of the user as Member or Casual. We inspect various algorithms to achieve this goal and compare their performance.

---

## **3 Hardware and Software Requirements**

### **3.1 Hardware Requirements**

1.500 GB HDD

2.4GB RAM

3.Monitor

4.Keyboard

### **3.2 Software Requirements**

1.64 bit Open Source Operating System like Ubuntu 18.04

2.Python 3

3.Google Colab

4.Different Libraries

5.Libraries like sklearn, pandas, matplotlib

---

## 4 INTRODUCTION

The data includes:

1. Duration – Duration of trip
2. Start Date – Includes start date and time
3. End Date – Includes end date and time
4. Start Station – Includes starting station name and number
5. End Station – Includes ending station name and number
6. Bike Number – Includes ID number of bike used for the trip
7. Member Type – Indicates whether user was a "registered" member (Annual Member, 30-Day Member or Day Key Member) or a "casual" rider (Single Trip, 24-Hour Pass, 3-Day Pass or 5-Day Pass)

This data has been processed to remove trips that are taken by staff as they service and inspect the system, trips that are taken to/from any of "test" stations at warehouses and any trips lasting less than 60 seconds (potentially false starts or users trying to re-dock a bike to ensure it's secure).

We first preprocess the data. The Data fields are used to extract year, month, day, hour, week, minute and second. We add this as features to our dataset. Out of this we keep only month and hour field as rest fields have a uniform distribution across entire dataset.

We perform one hot encoding of the remaining data fields and use Random Forest Classifier with default parameters as our classification algorithm. We use train test split of 60-40 and report a classification accuracy of 90 %.

---

## **5 OBJECTIVE**

- To analyse trip history dataset
- To predict the class of the user from given dataset..

---

## 6 Scope

We select only the 2010 file of records for our analysis. This is because the size of whole dataset makes it difficult to run the model on limited memory, currently we are making use of 115,000 records on the datasets to make our models work.



---

## 7 System Architecture

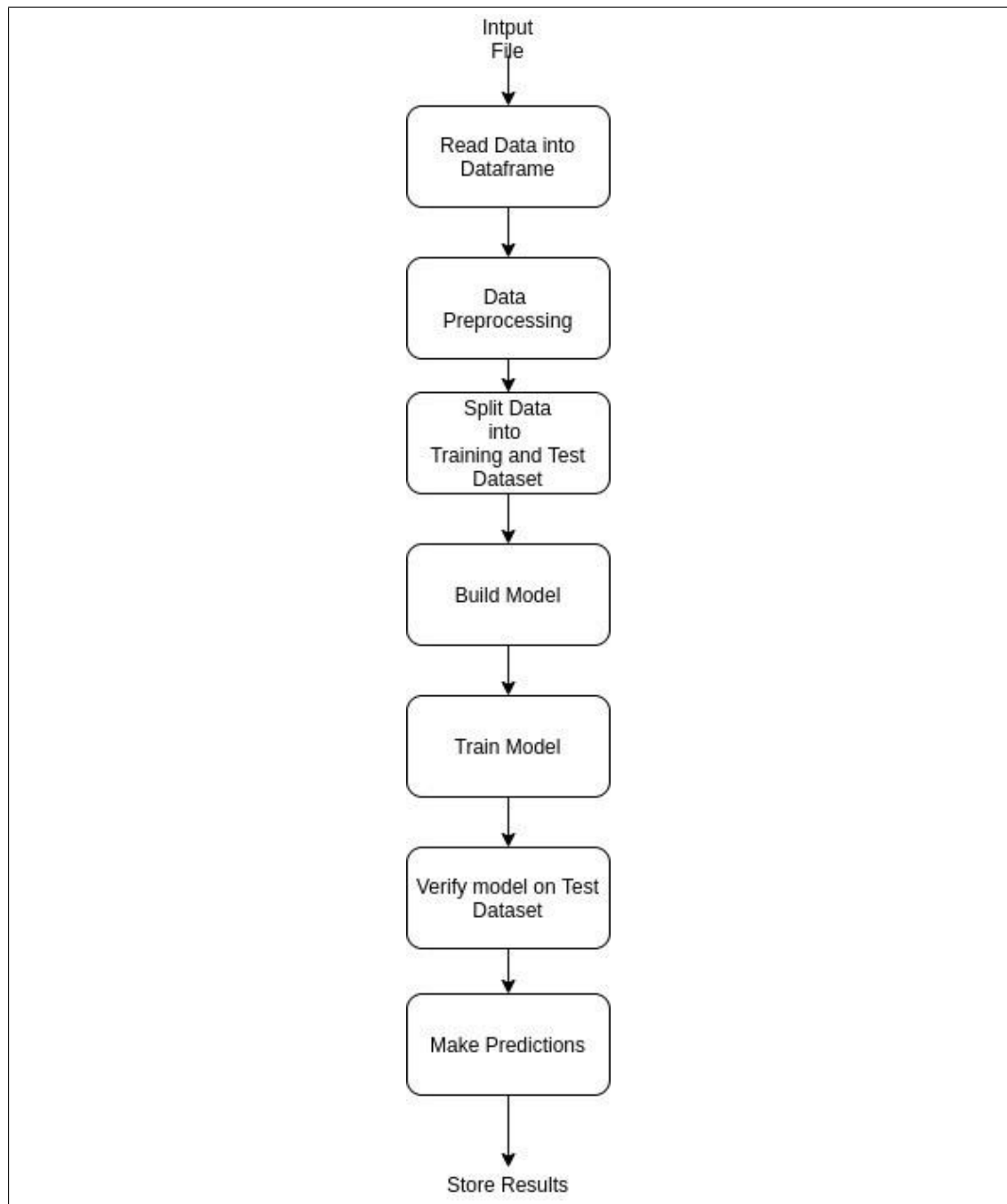


Figure 1: System Architecture

---

## 8 Result

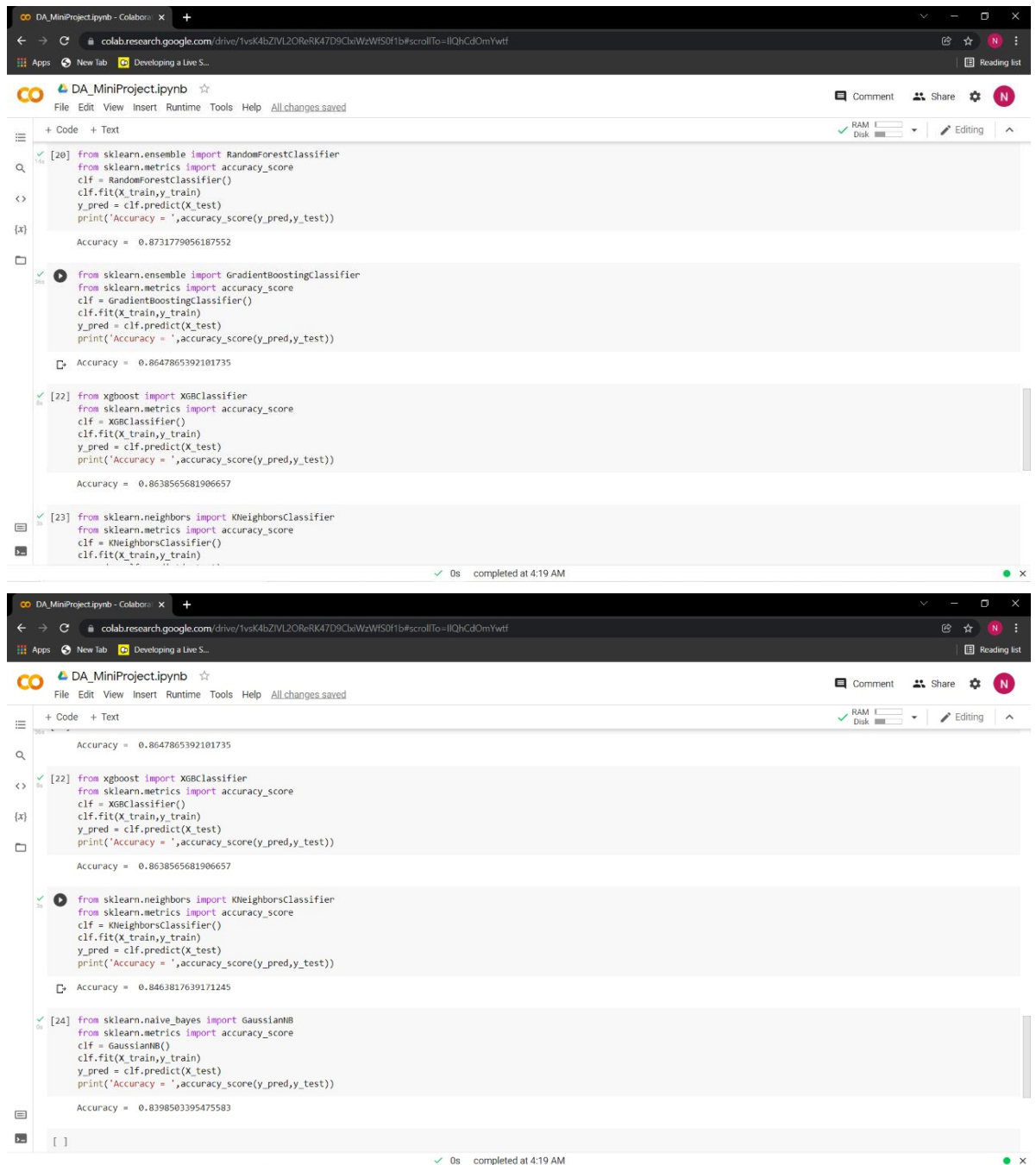
The Cross Validation Scores for Various models are:

Model	Cross-Validation Score
DecisionTree	0.7924939083293684
RandomForest	0.8701789293077843
GradientBoostingClassifier	0.8647865392101735
XGBoost	0.8642098129965253
KNeighborsClassifier	0.8464033911501363
GaussianNB	0.8398503395475583
SVC	0.8501232752281673
LinearSVC	0.8334054241100394

Table 1: Cross Validation Scores for vaious Models

We see that Random Forest 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 87 %.

## 9 Output



The image displays two screenshots of a Jupyter Notebook interface, showing the output of different machine learning classifiers. The notebook is titled "DA\_MiniProject.ipynb" and is running on a Google Colab environment. The first screenshot shows the output for Random Forest, Gradient Boosting, and XGBoost classifiers. The second screenshot shows the output for XGBoost, K-Nearest Neighbors, and Naive Bayes classifiers.

**First Screenshot Output:**

```
[20] from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8731779056187552
```

```
[21] from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
clf = GradientBoostingClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8647865392101735
```

```
[22] from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
clf = XGBClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8638565681906657
```

```
[23] from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
clf = KNeighborsClassifier()
clf.fit(X_train,y_train)
```

**Second Screenshot Output:**

```
[22] from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
clf = XGBClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8638565681906657
```

```
[23] from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
clf = KNeighborsClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8463817639171245
```

```
[24] from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
clf = GaussianNB()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8398503395475583
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
clf = SVC()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8501232752281673

from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
clf = LinearSVC()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.8184610060988797
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
ConvergenceWarning,

[19]: from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_pred,y_test))

Accuracy = 0.7967905186218477

[20]: from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier()
```

0s completed at 4:19 AM

---

## 10 Conclusion

We presented classification of trip history dataset to predict the class of user using Random Forest Classifier. We report a classification accuracy of 87%.

---

## References

- [1] <https://www.capitalbikeshare.com/system-data>
- [2] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>