A MINI PROJECT REPORT

On

TRIP HISTORY

Submitted by

Abhilakshya Agarwal (171520001)

Anu Dhull (171510013)

Aryan Sharma (171520006)

Ashree Verma (171520007)

Krishna Bansal (171720016)

Pralabh Saxena (171510038)

Department of Computer Engineering &Applications

Institute of Engineering &Technology

GLA University, Mathura

Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

We hereby declare that the work which is being presented in the Mini Project "**Trip History",** in fulfillment of the requirements for Mini-Project, is an authentic record of our own work carried under the supervision of **Mr. Ashutosh Shankhdhar, Assistant Professor, GLA University, Mathura**.

Sign_____

Abhilakshya Agarwal

Sign_____

Anu Dhull

Sign_____

Aryan Sharma

Sign_____

Ashree Verma

Sign_____

Krishna Bansal

Sign_____

Pralabh Saxena

Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

CERTIFICATE

This is to certify that the project entitled **"*Trip History.*"** carried out in Mini Project is a bonafide work done by ***Abhilakshya Agarwal (171520001), Anu Dhull (171510013), Aryan Sharma (171520006), Ashree Verma (171520007), Krishna Bansal(171520016) and Pralabh Saxena(171510038)*** and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Sign_____

Mr. Ashutosh Shankhdhar

(Mentor)

Sign_____

Mr. Saurabh Anand

(IBM Coordinator)

Date……

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to **Prof. (Dr.) Anand Singh Jalal,** Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe special debt of gratitude to **Mr. Ashutosh Shankhdhar,** Assistant Professor Department of CEA, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Abhilakshya Agarwal

Anu Dhull

Aryan Sharma

Ashree Verma

Krishna Bansal

Pralabh Saxena

ABSTRACT

This dataset comes from a bike sharing service in the United States. This dataset requires you to exercise your pro data munging skills. The data is provided quarter-wise from 2010 (Q4) onwards. Each file has 7 columns. It is a classification problem.

The objective is the exploration of the Capital Bikes Share Trip Dataset. We intend to explore the data, look at and plot the relationship between the variables. This is a classification issue that will make use of the Logistic Regression and different classification Algorithms to predict the likelihood that a certain combination of the Independent variables leads to a certain classification.

We shall explore a good number of hypothesis and deduce conclusions based on the output of the plots. Some feasible hypothesis includes but not limited to: There is a relationship between the duration one stays with a bike and their membership type There is a link between the End station and the Member Type Most members start at Station Number 31014 There is no link between the duration one stays with a bike and an End

Table of Contents

Chapter 1

Introduction

1.1 Overview

This dataset comes from a bike sharing service in the United States. This dataset requires to exercise the pro data mugging skills. The data is provided quarter-wise from 2010 (Q4) onwards. Each file has 7 columns. It is a classification problem.

The objective is the exploration of the Capital Bikes Share Trip Dataset. We intend to explore the data, look at and plot the relationship between the variables. This is a classification issue that will make use of the Logistic Regression Algorithm to predict the likelihood that a certain combination of the Independent variables lead to a certain classification.

We shall explore a good number of hypothesis and deduce conclusions based on the output of the plots.

There is a relationship between the duration one stays with a bike and their membership type. There is a link between the End station and the Member Type. Most members start at Station Number 31014 There is no link between the duration one stays with a bike and an End station number.

The data is provided according to the capital BikeShare Data License agreement.

Trip History Data
Each quarter, we publish downloadable files of capital Bikeshare trip data. The data includes:
Duration – Duration of the 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 Number – includes ID number of bike used for the trip
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 Staffs as they service and inspect the system, trips that are taken to/from any of our "test" station at our 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).

Dataset link : https://s3.amazonaws.com/capitalbikeshare-data/index.html

1.2 Objective

This dataset comes from a bike sharing service in the United States. This dataset requires to exercise the pro data mugging skills. The data is provided quarter-wise from 2010 (Q4) onwards. Each file has 7 columns. It is a classification problem.

The objective is the exploration of the Capital Bikes Share Trip Dataset. We intend to explore the data, look at and plot the relationship between the variables. This is a classification issue that will make use of the Logistic Regression Algorithm to predict the likelihood that a certain combination of the Independent variables lead to a certain classification.

Where do Capital Bikeshare riders go? When do they ride? How far do they go? Which stations are most popular? What days of the week are most rides taken on? We have heard all of these questions since we launched in 2010, and we are glad to provide the data that shows you the answers from our first trips to today.

This data has been processed to remove trips that are taken by Staffs as they service and inspect the system, trips that are taken to/from any of our "test" station at our 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).

Chapter 2

Software Development Tools

2.1 Tools Used

Python – The project is developed in python.

Jupyter Notebook – Jupyter Lab is a web-based interactive development environment for Jupyter notebooks, code, and data. Jupyter Lab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plug-ins that add new components and integrate with existing ones.

Python IDLE - IDLE is integrated development environment (IDE) for editing and running Python 2.x or Python 3 programs. The IDLE GUI (graphical user interface) is automatically installed with the Python interpreter. IDLE was designed specifically for use with Python.

2.2 Modules and their functionalities

NumPy:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

A powerful N-dimensional array object

sophisticated (broadcasting) functions

tools for integrating C/C++ and Fortran code

useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD License, enabling reuse with few restrictions.

Pandas:

Python has long been great for data munging and preparation, but less so for data analysis and modeling. *Pandas* helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without

having to switch to a more domain specific language like R.

Python has long been great for data munging and preparation, but less so for data analysis and modeling. *panda*shelps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.

Matplotlib - Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Skicit-learn - **Scikit-learn** (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.3 Requirements

1) Hardware Requirements

• Minimum 4 GB RAM, and

• Minimum i3 processor.

2) Software Requirements

• Any windows based operating system,

• Python,

•Jupyter

Chapter 3
Libraries

Pandas

Pandas is a Python pa**c**kage providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language**.** It is already well on its way toward this goal.

Functions

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Hierarchical labeling of axes (possible to have multiple labels per tick)

- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Installation

The best way to get pandas is via conda

Command - condainstallpandas

Packages are available for all supported python versions on Windows, Linux, and MacOS.

Wheels are also uploaded to PyPI and can be installed with

Command - pip install pandas

Matplotlib

Matplotlib.pyplotis a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the *axes*part of a figure and not the strict mathematical term for more than one axis).

Installation:

Matplotlib and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

Commands:

```
python-mpipinstall-Upip
python-mpipinstall-Umatplotlib
```

Functions:

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:
a powerful N-dimensional array object
sophisticated (broadcasting) functions
tools for integrating C/C++ and Fortran code
useful linear algebra, Fourier transform, and random number capabilities
Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Installation

NumPy can be installed via following command:

Command - pip install numpy

Scikit-Learn

scikit-learn is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface.

Scikit-learn is largely written in Python, and uses numpy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in python to improve performance. Support vector machines are implemented by a python wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, pandas dataframes, scipy, and many more.

Scikit-learn (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning

library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Important features of scikit-learn:

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

Some popular groups of models provided by scikit-learn include:

- **Clustering**: for grouping unlabeled data such as KMeans.
- **Cross Validation**: for estimating the performance of supervised models on unseen data.
- **Datasets**: for test datasets and for generating datasets with specific properties for investigating model behavior.
- **Dimensionality Reduction**: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
- **Ensemble methods**: for combining the predictions of multiple supervised models.
- **Feature extraction**: for defining attributes in image and text data.
- **Feature selection**: for identifying meaningful attributes from which to create supervised models.
- **Parameter Tuning**: for getting the most out of supervised models.
- **Manifold Learning**: For summarizing and depicting complex multi-dimensional data.
- **Supervised Models**: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

Installation

Command – `pip install -U scikit-learn`

Chapter 4
Implementation

4.1 IMPORT LIBRARIES

```
*numpy and pandas*
import pandas as pd
import numpy as np

*data preparation*
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

*machine learning algorithm*
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

*Neural Network*
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler

*result evaluation*
from sklearn import metrics
from sklearn.metrics import classification_report
```

Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.linear_model import LogisticRegression
```

4.2 DATA COLLECTING

Merging of all data into one csv file:

```
[ ]: import pandas as pd
     from glob import glob

[ ]: df_final = sorted(glob('dataset/*-capitalbikeshare-tripdata.csv'))
     df_final

[ ]: alldata = pd.concat((pd.read_csv(file).assign(filename = file)
                for file in df_final),ignore_index=True)
     alldata.head()

[ ]: alldata.to_csv('dataset/final.csv',index=None)

[ ]: alldata.tail()
```

Importing data to Jupyter Notebook

```
: data = pd.read_csv("C:/Users/Aryan Sharma/OneDrive/Desktop/dataset/final.csv")

: data.head()
```

	Duration	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member type
0	1012	2010-09-20 11:27:04	2010-09-20 11:43:56	31208	M St & New Jersey Ave SE	31108	4th & M St SW	W00742	Member
1	61	2010-09-20 11:41:22	2010-09-20 11:42:23	31209	1st & N St SE	31209	1st & N St SE	W00032	Member
2	2690	2010-09-20 12:05:37	2010-09-20 12:50:27	31600	5th & K St NW	31100	19th St & Pennsylvania Ave NW	W00993	Member
3	1406	2010-09-20 12:06:05	2010-09-20 12:29:32	31600	5th & K St NW	31602	Park Rd & Holmead Pl NW	W00344	Member
4	1413	2010-09-20 12:10:43	2010-09-20 12:34:17	31100	19th St & Pennsylvania Ave NW	31201	15th & P St NW	W00883	Member

4.3 DATA CLEANING AND PREPROCESSING:

```
data['Member type'].value_counts()
```

```
Member      20726661
Casual       5360404
Unknown           58
Name: Member type, dtype: int64
```

Target variable is **Member type,** so we need data of Member type to be accurate so that out model can train properly. So we remove all rows having the missing value or unknown values in Member type column.

[3]:
```
# we did this because we need to filter out the unknown members type
data = data[(data['Member type']=='Casual') | (data['Member type']=='Member')]
```

[15]:
```
data['Member type'].value_counts()
```

[15]:
```
Member    20726661
Casual     5360404
Name: Member type, dtype: int64
```

We have

Duration – Duration of the 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 Number – includes ID number of bike used for the trip

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)

We don’t want attribute – Start date , end date inspite of these two attribute Duration can be use.

Bike Number is also of no use as any bike can be use from any station.

So columns Duration , Start Station number, End station number and target column Member type is filtered out from main data.

```
[4]: x = data.iloc[:,[0,3,5]].values
     y = data.iloc[:,-1].values
```

Our target attribute is categorical , we cant train it in this form so we tranform or encode it in binary form 0 as Member and 1 as Casual.

```
In [5]: # Transforming the type into an encoded form because it is categorical
        le = LabelEncoder()
        y = le.fit_transform(y.flatten())
```

```
In [18]: print(y)
         [1 1 1 ... 1 1 1]
```

4.4 DATA SPLITTING

Dividing the given data into testing and training dataset so that we can train our model on train dataset and testing dataset can be used for Evaluation purpose.

```
: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=0)
```

4.5 TRAINING MACHINE LEARNING MODELS

(A) (i) Decision Tree Classifier

Training model:

```
# Decision Tree Classifier

tree = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
tree.fit(X_train, y_train)
```

Predicting :

```
y_pred = tree.predict(X_test)
```

Evalutaion of Decision Tree Model:

```
# Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 84.0%
             precision    recall  f1-score   support

          0       0.65      0.48      0.55   1608003
          1       0.87      0.93      0.90   6218117

avg / total       0.83      0.84      0.83   7826120
```

(ii) Decision tree classifier with other attribute:

Training model:

```
#Decision Tree Classifier with different attributes
tree_two = DecisionTreeClassifier(criterion="entropy", max_depth=11)
tree_two.fit(X_train, y_train)
```

Predicting :

```
y_pred = tree_two.predict(X_test)
```

Evalution:

```
# Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 86.9%
             precision    recall  f1-score   support

          0       0.76      0.53      0.62   1608003
          1       0.89      0.96      0.92   6218117

avg / total       0.86      0.87      0.86   7826120
```

(B) K- Nearest Neighbors Classifier

Training:

```
# K- Nearest Neighbors
neighbor = KNeighborsClassifier(10)
neighbor.fit(X_train, y_train)
```

Predicting:

```
y_pred = neighbor.predict(X_test)
```

Evaluation:

```
# Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 86.8%
             precision    recall  f1-score   support

          0       0.71      0.61      0.65   1608003
          1       0.90      0.93      0.92   6218117

avg / total       0.86      0.87      0.86   7826120
```

(C) Random Forest Classifier

Training:

```
|: # Random Forest
rfc=RandomForestClassifier(n_estimators=15,criterion='entropy',random_state=0)
rfc.fit(X_train,y_train)
```

```
|: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=15, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Predicitng:

```
: y_pred = rfc.predict(X_test)
```

Evaluation:

```
# Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 86.3%
              precision    recall  f1-score   support

           0       0.69      0.61      0.65   1608003
           1       0.90      0.93      0.92   6218117

    accuracy                           0.86   7826120
   macro avg       0.80      0.77      0.78   7826120
weighted avg       0.86      0.86      0.86   7826120
```

(D) Logistic Regression

Training:

```
# Logisitic Regression
logis = LogisticRegression(random_state = 0)
logis.fit(X_train, y_train)
```

Predicting:

```
y_pred = logis.predict(X_test)
```

Evaluation:

```
#Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 83.8%
             precision    recall  f1-score   support

          0       0.83      0.27      0.40   1608003
          1       0.84      0.99      0.91   6218117

avg / total       0.84      0.84      0.80   7826120
```

(E) Neural Networks:

Preparation of data:

```
#Neural Network

#converting the values from integer to floar
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
C:\Users\acer\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning: Data with input dty
pe int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

Training:

```
neuralnet = MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=(5, 3), random_state=1)
neuralnet.fit(X_train, y_train)
```

```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(8, 1), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

Prediction:

```
: y_pred = neuralnet.predict(X_test) # performing prediction
```

Evaluation:

```
#Accuracy
score = metrics.accuracy_score(y_test, y_pred)
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 86.0%
             precision    recall  f1-score   support

          0       0.77      0.46      0.57   1608003
          1       0.87      0.96      0.92   6218117

avg / total       0.85      0.86      0.85   7826120
```

Chapter 5
Conclusion

The new bike sharing networks have created a significant volume of data on bike use, providing researchers with an important tool for studying trends of human movement in urban environments. In this paper, we propose classification methods to infer bike trip patterns directly from the public station feeds, classified that bike members are members or casual bikers.

As for the applicability of the models tested, the analysis of this dataset has been supervised and the labels of the different categories may not always be accessible. So, we designed a system to take together seven factors and implemented five models. Because each model predicted a different result because of the tree structure or hyper parameters we set, stacked model used these results to train new models that minimized error. By re-evaluating and re-assuring the predicted results, this improved efficiency.

With Random Forest's support, we're not able to achieve good precision, F1 Score and remember. But we do obtain strong results with the aid of other models. Decision Tree with Precision is the strongest one achieved 86.9 percent.

Chapter 6

References

[1] Udry J. R., "The nature of gender," Demography, vol. 31, pp. 561-573, 1994.

[2] Demirkus M., Garg K., and Guler S., "Automated person categorization for video surveillance using soft biometrics," in Biometric Technology for Human Identification VII, 2010, p. 76670P.

[3] Jain A. K., Ross A., and Prabhakar S., "An introduction to biometric recognition," IEEE Transactions on circuits and systems for video technology, vol. 14, pp. 4-20, 2004.

[4] Paluchamy M., Suganya D., and Ellammal S., "Human gait-based gender classification using various transformation techniques," IJRCCT, vol. 2, pp. 1315-1321, 2013.

[5] Gnanasivam P. and Muttan S., "Gender classification using ear biometrics," in Proceedings of the Fourth International Conference on Signal and Image Processing 2012 (ICSIP 2012), 2013, pp. 137-148.

[6] Amayeh G., Bebis G., and Nicolescu M., "Gender classification from hand shape," in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, 2008, pp. 1-7.

[7] Hoffmeyer-Zlotnik J. H. and Wolf C., Advances in cross-national comparison: A European working book for demographic and socio-economic variables: Springer Science & Business Media, 2003.

[8] Rakesh K., Dutta S., and Shama K., "Gender Recognition using speech processing techniques in LABVIEW," International Journal of Advances in Engineering & Technology, vol. 1, pp. 51-63, 2011.

[9] Khan S. A., Ahmad M., Nazir M., and Riaz N., "A comparative analysis of gender classification techniques," MiddleEast Journal of Scientific Research, vol. 20, pp. 1-13, 2014.

[10] Mäkinen E. and Raisamo R., "An experimental comparison of gender classification methods," pattern recognition letters, vol. 29, pp. 1544-1556, 2008.

[11] Chen G., Feng X., Shue Y.-L., and Alwan A., "On using voice source measures in automatic gender classification of children's speech," in Eleventh Annual Conference of the International Speech Communication Association, 2010.

[12] Meena K., Subramaniam K. R., and Gomathy M., "Gender classification in speech recognition using fuzzy logic and neural network," Int. Arab J. Inf. Technol., vol. 10, pp. 477-485, 2013.

[13] The Harvard-Haskins Database of RegularlyTimed Speech, http://www.nsi.edu/~ani/download.html

[14] Jayasankar T., Vinothkumar K., and Vijayaselvi A., "Automatic Gender Identification in Speech Recognition by Genetic Algorithm," Appl. Math, vol. 11, pp. 907-913, 2017.

[15] oxForge Speech Corpus, http://www.repository.voxforge1.org/downloads/Sp eechCorpus/Trunk/Audio/ Main/8kHz_16bit