Supervised Machine Learning

(Understanding and Predicting Property Maintenance Fines)

Report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Computer Science & Engineering

By

MANAN CHHABRA

(05515002718)

Maharaja Surajmal Institute of Technology

Affiliated to Guru Gobind Singh Indraprastha University

Janakpuri, New Delhi-110058

2018-2022

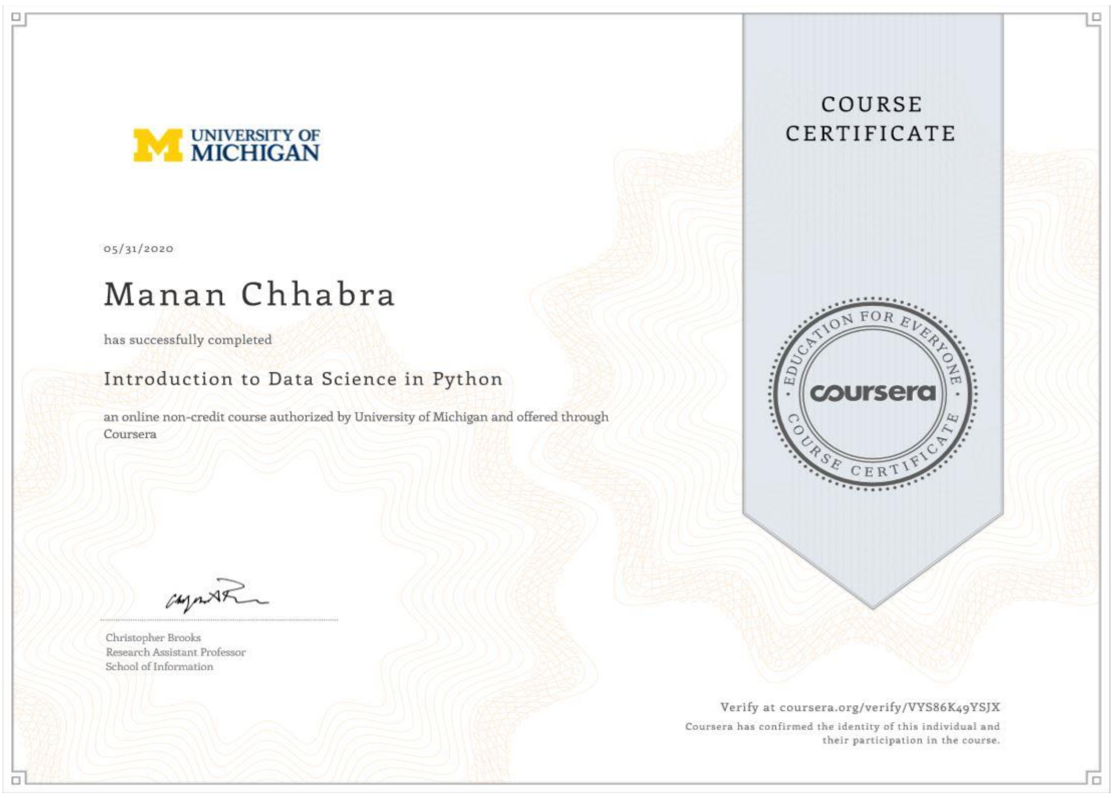
Certificate by Student

This is to certify that Report entitled "**Supervised Machine Learning (Understanding and Predicting Property Maintenance Fines)**" which is submitted in partial fulfilment of the requirement for the award of degree B.Tech. in Computer Science and Engineering to MSIT, GGSIP University, Dwarka, Delhi comprises only my original work and due acknowledgment has been made in the text to all other material used.

Date: 28 Sept 2020 **Name of Student**

MANAN CHHABRA
(05515002718)

Certificate by Organization



UNIVERSITY OF MICHIGAN

COURSE CERTIFICATE

05/31/2020

Manan Chhabra

has successfully completed

Introduction to Data Science in Python

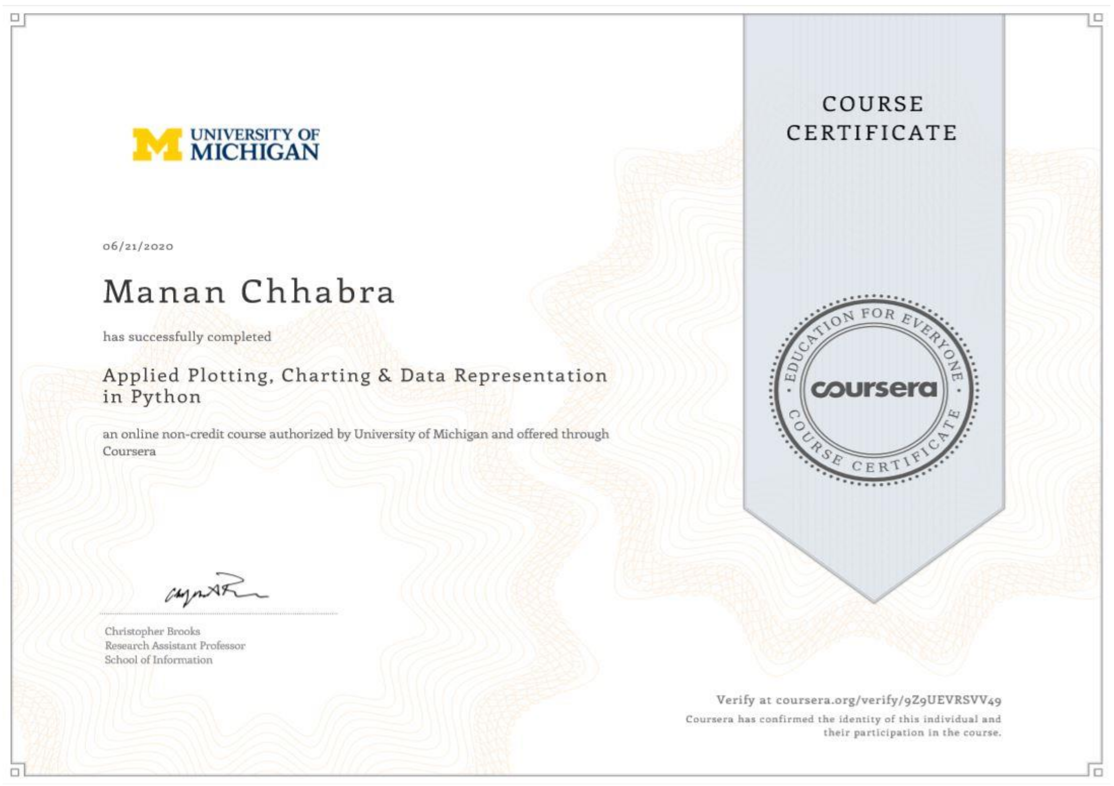
an online non-credit course authorized by University of Michigan and offered through Coursera

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

[illegible]

Christopher Brooks
Research Assistant Professor
School of Information

Verify at coursera.org/verify/VYS86K49YSJX
Coursera has confirmed the identity of this individual and their participation in the course.



UNIVERSITY OF MICHIGAN

COURSE CERTIFICATE

06/21/2020

Manan Chhabra

has successfully completed

Applied Plotting, Charting & Data Representation in Python

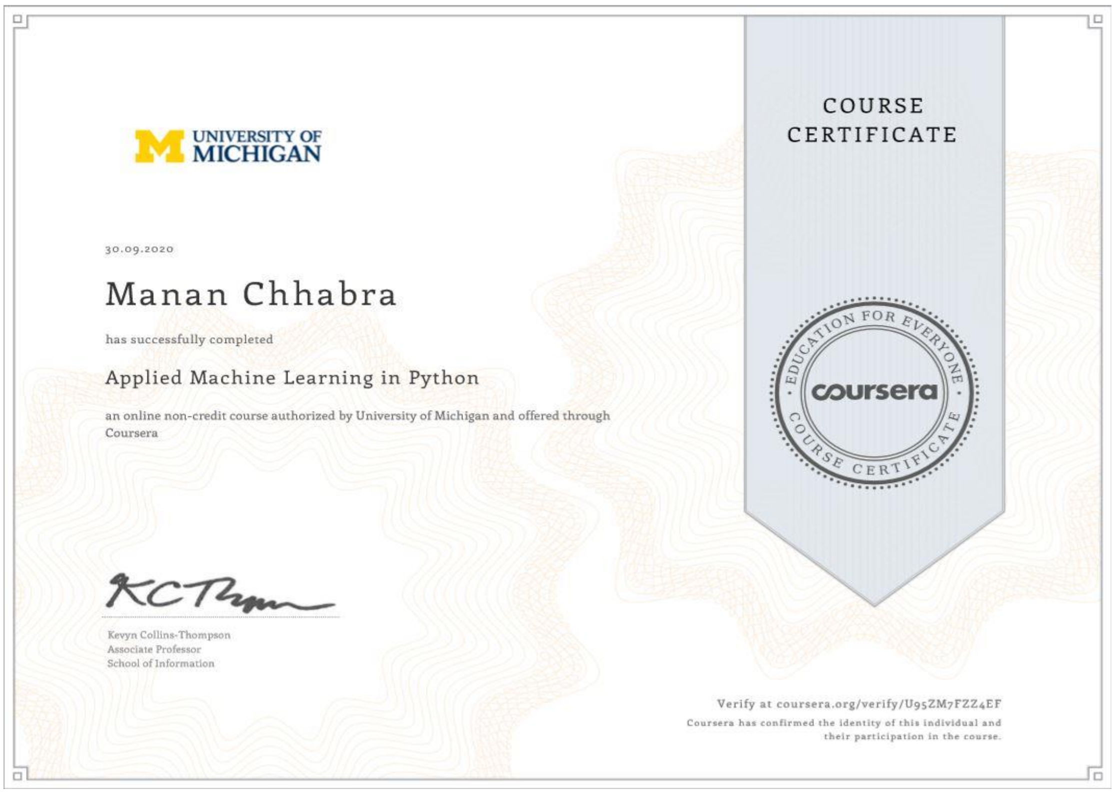
an online non-credit course authorized by University of Michigan and offered through Coursera

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

[illegible]

Christopher Brooks
Research Assistant Professor
School of Information

Verify at coursera.org/verify/9Z9UEVRSVV49
Coursera has confirmed the identity of this individual and their participation in the course.



UNIVERSITY OF MICHIGAN

COURSE CERTIFICATE

30.09.2020

Manan Chhabra

has successfully completed

Applied Machine Learning in Python

an online non-credit course authorized by University of Michigan and offered through Coursera

EDUCATION FOR EVERYONE coursera COURSE CERTIFICATE

[illegible]

Kevyn Collins-Thompson
Associate Professor
School of Information

Verify at coursera.org/verify/U95ZM7FZZ4EF
Coursera has confirmed the identity of this individual and their participation in the course.

ACKNOWLEDGEMENT

First and foremost, we would like to thank God Almighty for making us capable and fortunate enough that we have been given a chance to pursue an innovative and knowledgeable course like B.Tech. from one of the most reputed universities in our country.

We are very grateful to our director **Dr. K.P. Chaudhary** for giving us the beautiful opportunity of working on this project and to the other faculty members, related to the project, for providing their valuable time and attention to help us in the project.

We are also thankful to **Dr. Sapna Malik** for her most valuable and significant guidance in our understanding of the project and in elaborating our view of studying the project's details and getting the right vision for its implementation.

We would also like to thank our Head of Department, **Dr. Rinky Dwivedi**, who provided a wide vision of the scope of the project which was very important for us.

At last, we would like to thank all the faculty members and lab in-charge who were directly or indirectly involved in the development of this project by providing much-needed support and cooperation.

ABSTRACT

This report presents the five tasks, completed while making this project by the knowledge gained from the internship done at University of Michigan, and are listed below:

i. Collecting the data of the Blight violations that are issued by the city to individuals who allow their properties to remain in a deteriorated condition.
ii. Understanding the data and using it to predict whether the blight fines will be paid or not.
iii. Cleaning and analysing the data to form a dataset.
iv. Deploying a model to predict the labels based on the patterns recognized by the model
v. Find the accuracy and confusion matrix to get a rough idea to find which model predicts better.

As people in the City of Detroit leave their properties in deteriorated conditions, it is a very tedious task to predict whether the fine will be paid or not. This project aims to predict the number of citizens who will pay the fine, with the most accurate results by deploying various models from the in-built scikit learn library.

Models like scikit learn helps to ease out our work as they have in-built functionalities which help us to achieve our goal. If further developed, this idea can turn into a useful product for mankind.

All the tasks are completed successfully, and the project is completed and is running with no error found.

Table of Contents

Contents

Chapter 1: Introduction

1.1 Needs and Objectives:

Needs:

The need for this project is to help solve one of the most pressing problems in the City of Detroit facing Detroit - blight. Blight violations are issued by the city to individuals who allow their properties to remain in a deteriorated condition. Every year, the city of Detroit issues millions of dollars in fines to residents and every year, many of these fines remain unpaid. Enforcing unpaid blight fines is a costly and tedious process, so the city wants to know: how can we increase blight ticket compliance?

Objectives:

i. Collecting the data of the Blight violations that are issued by the city to individuals who allow their properties to remain in a deteriorated condition.
ii. Understanding the data and using it to predict whether the blight fines will be paid or not.
iii. Cleaning and analysing the data to form a dataset.
iv. Deploying a model to predict the labels based on the patterns recognized by the model.
v. Find the accuracy and confusion matrix to get a rough idea to find which model predicts better.

1.2 Methodology

The methodology, which was used to make this application and accomplish all the objectives are as follows:

1. Phase -1: Initiation

The initiation of a project or a system begins when a need or a problem is identified. In this project, the problem was identified as a real life-based problem of difficulty in finding the appropriate tech YouTube videos.

2. Phase -2: Data Collection

In this phase, the data was collected, as soon as the problem was identified. The data was needed to be stored and manipulated according to the developer. The most difficult part of this phase was to maintain the quality of data that was being collected as data is the key factor to save the model from overfitting or underfitting.

3. Phase -3: Research and Analysis

The scikit learn library in python consists of a large number of models that can help us achieve our goal but differ in the results. Further, we need to research various other libraries that can help us to contribute to easing our work. Since a large number of libraries are present of which no data scientist knows about all these libraries. We needed to look for various libraries and study their various syntax and advantages that we can use to fulfill our requirements.

4. Phase -4: Development

In this phase, the detailed specifications studied during the research phase are translated into working algorithms. The required libraries are installed if not present in the system. The data collected is converted to an excel file.

5. **Phase -5: Pre-Processing Text**

The data is imported as a dataframe and various operations are performed like pre-processing, tokenization, vectorization, etc.

6. **Phase -6: Testing**

In this phase, various machine learning estimators/models are tested out, their results compared, and the best model is selected. The hyper-parameters are tuned and then, the best is selected that give the best results.

1.3 Software Used:

- For Programming - Jupyter Notebook, Anaconda Navigator
- Software Required - Jupyter Notebook
- Language Used - Python 3

Hardware Used

● Hard disk	- 100 GB
● Ram	- 8 GB
● OS	- Windows/Mac OS

Hardware Required

● Hard disk	- 100 MB
● Ram	- 512 MB
● OS	- Platform Independent

Software Required: Jupyter Notebook, Anaconda Navigator

2.2.1 Anaconda Navigator:

Anaconda is a scientific Python distribution. Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. It has no IDE of its own. The default IDE bundled with Anaconda is Spyder which is just another Python package that can be installed even without Anaconda.

Anaconda bundles a whole bunch of Python packages that are commonly used by people using Python for scientific computing and/or data science. It provides a single download and an install program/script that installs all the packages in one go. Alternatively, one can install Python first and then individually install all the required packages using pip. Additionally, it provides its own package manager (conda) and package repository. But it allows the installation of packages from PyPI using pip if the package is not in Anaconda repositories. It is especially good if you are installing on Microsoft Windows as it can easily install packages that would otherwise require you to install C/C++ compilers and libraries if you were using pip. It is certainly an added advantage that conda, in addition to being a package manager, is also a virtual environment manager allowing you to install independent development environments and switch from one to the other (similar to virtualenv).

There is a minimal Anaconda Python without all the packages, called Miniconda. After installing miniconda, you can use conda and install only those scientific packages that you wish and avoid a bloated installation.

2.2.2 Jupyter Notebook:

Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use, and released under the liberal terms of the modified BSD license.

It is a web-based interface that allows for rapid prototyping and sharing of data-related projects. It works with many kernels (this is the name given to the code env that it can run) The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Jupyter is developed in the open on GitHub, through the consensus of the Jupyter community. Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala. Notebooks can be shared with others using email, Dropbox, GitHub, and the Jupyter Notebook Viewer.

2.2.2 PYTHON 3:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum from 1985 to 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called ëMonty Pythonís Flying Circusí and not after Python-the snake.

We chose Python because of the following features:

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can sit at a Python prompt and interact with the

  interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the

  beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.
- Scalable − Python provides a better structure and support for large programs than shell scripting.
- A broad standard library − Python's bulk of the library is very portable and

  cross-platform compatible with UNIX, Windows, and Macintosh.

1.4 About the Organization

The mission of the University of Michigan is to serve the people of Michigan and the world through pre-eminence in creating, communicating, preserving and applying knowledge, art, and academic values, and in developing leaders and citizens who will challenge the present and enrich the future.

The University of Michigan was founded in 1817 as one of the first public universities in the nation. The University, which is based in Ann Arbor, Michigan, has 260 Degree programs in 19 schools and colleges offering a tremendous academic breadth and opportunity for discovery. Michigan has one of the largest alumni networks in the world with more than 540,000 living alumni.

The University of Michigan created the Office of Academic Innovation to shape the future of learning and redefine public residential education at a 21st century research university by unlocking new opportunities and enabling personalized, engaged, and lifelong learning for the U-M community and learners around the world. Through Coursera, the University furthers its commitment to shaping the future of learning.

Chapter 2: Project Design

2.1 Introduction

Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs that can be used to achieve the desired project goals.
In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.

2.2 Problem Description:

i. Collecting the data of the Blight violations that are issued by the city to individuals who allow their properties to remain in a deteriorated condition.
ii. Understanding the data and using it to predict whether the blight fines will be paid or not.
iii. Cleaning and analysing the data to form a dataset.
iv. Deploying a model to predict the labels based on the patterns recognized by the model.
v. Find the accuracy and confusion matrix to get a rough idea to find which model predicts better.

2.3 Machine Learning Pipeline

A machine learning pipeline can be broken down into three major steps. Data collection, data modeling, and deployment. All influence one another. You may start a project by collecting data, model it, realize the data you collected was poor, go back to collecting data, model it again, find a good model, deploy it, find it doesn't work, make another model, deploy it, find it doesn't work again, go back to data collection. It's a cycle. How you collect data will depend on your problem. Modeling refers to using a machine learning algorithm to find insights within your collected data.

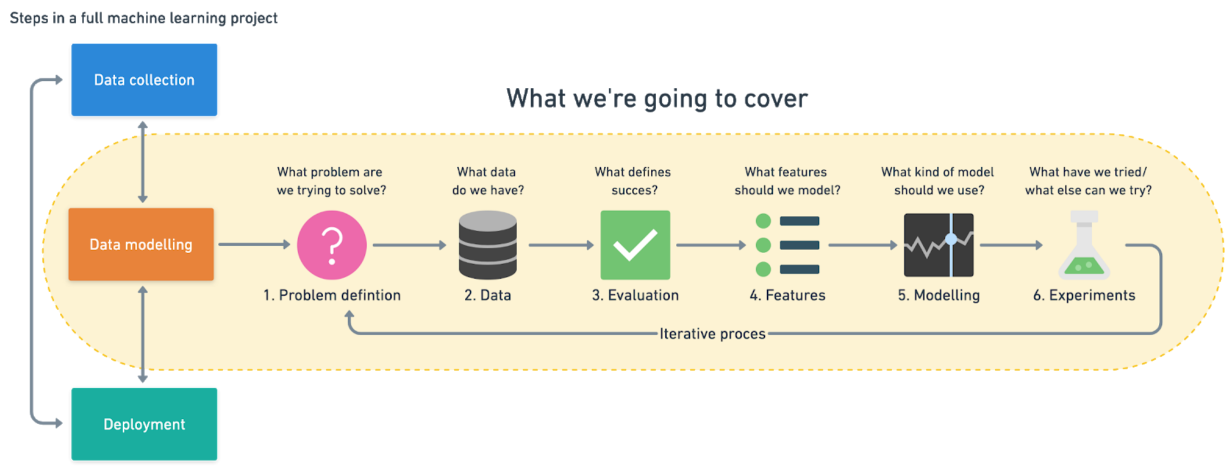

Steps in a full machine learning project
Data collection
What we're going to cover
Data modelling
What problem are we trying to solve?
1. Problem defintion
What data do we have?
2. Data
What defines succes?
3. Evaluation
What features should we model?
4. Features
What kind of model should we use?
5. Modelling
What have we tried/ what else can we try?
6. Experiments
Iterative proces
Deployment

Fig: Block diagram representing steps in a full machine learning project

What's the difference between a normal algorithm and a machine learning algorithm?

Like a cooking recipe for your favorite chicken dish, a normal algorithm is a set of instructions on how to turn a set of ingredients into that honey mustard masterpiece. What makes a machine learning algorithm different is instead of having the set of instructions, you start with the ingredients and the final dish ready to go. The machine-learning algorithm then looks at the ingredients and the final dish and works out the set of instructions. There are many different types of machine learning algorithms and some perform better than others on different problems. But the premise remains, they all have the goal of finding patterns or sets of instructions in data.

2.3.1 Problem Definition

To help decide whether or not your business could use machine learning, the first step is to match the business problem you're trying to solve a machine learning problem.

The four major types of machine learning are supervised learning, unsupervised learning, transfer learning, and reinforcement learning (there's semi-supervised as well). The three most used in business applications are supervised learning, unsupervised learning, and transfer learning.

2.3.1.1 Supervised Learning

Supervised learning is called supervised because you have data and labels. A machine learning algorithm tries to learn what patterns in the data lead to the labels. The supervised part happens during training. If the algorithm guesses a wrong label, it tries to correct itself.

For example, if you were trying to predict heart disease in a new patient. You may have the anonymized medical records of 100 patients as the data and whether or not they had heart disease as the label.

2.3.1.2 Unsupervised Learning

Unsupervised learning is when you have data but no labels. The data could be the purchase history of your online video game store customers. Using this data, you may want to group similar customers so you can offer them specialized deals. You could use a machine-learning algorithm to group your customers by purchase history. After inspecting the groups, you provide the labels. There may be a group interested in computer games, another group who prefer console games, and another which only buy discounted older games. This is called clustering.

2.3.1.3 Transfer Learning

Transfer learning is when you take the information an existing machine learning model has learned and adjusted it to your problem. Training a machine learning model from scratch can be expensive and time-consuming. The good news is, you don't always have to. When machine learning algorithms find patterns in one kind of data, these patterns can be used in another type of data.

Let's say you're a car insurance company and wanted to build a text classification model to classify whether or not someone submitting an insurance claim for a car accident is at fault (caused the accident) or not at fault (didn't cause the accident). You could start with an existing text model, one which has read all of Wikipedia and has remembered all the patterns between different words, such as, which word is more likely to come next after another. Then using your car insurance claims (data) along with their outcomes (labels), you could tweak the existing text model to your problem.

2.3.2 Data

The data you have or need to collect will depend on the problem you want to solve. If you already have data, it will likely be in one of two forms. Structured or unstructured. Within each of these, you have static or streaming data.

- **Structured Data:** Data in the form of tables and rows. Example: A database of patient's records.
- **Unstructured Data:** Anything that cannot be put into rows and columns, images, audio files, natural text, etc. immediately.
- **Static Data:** Existing historical data that is unlikely to change in the future. Your customer purchase history is an example of this kind of data.
- **Streaming Data:** Data that is constantly updated, older records may be changed, newer records are constantly added.

2.3.3 Evaluation

You’ve defined your business problem in machine learning terms and you have data. Now define what defines success. There are different evaluation metrics for classification, regression, and recommendation problems. Which one you choose will depend on your goal. A 95% accurate model may sound pretty good for predicting who’s at fault in an insurance claim. But for predicting heart disease, you’ll likely want better results.

Other things you should take into consideration for classification problems:

- **False Negatives:** The model predicts negatives but is actually positive. In some cases, it may not be much of a problem but in cases like heart disease prediction, it’s very important.
- **False Positives:** The model predicts positive but is actually negative.
- **True Negatives:** The model predicts negative and is actually negative.
- **True Positives:** The model predicts positive and is actually positive.
- **Precision:** What proportion of positive predictions were actually correct? A model that produces no false positives has a precision of 1.0.
- **Recall:** What proportion of actual positives were predicted correctly? A model that produces no false negatives has a recall of 1.0.
- **F1 Score:** A combination of precision and recall. The closer to 1.0, the better.
- **Receiver operating characteristic (ROC) curve & Area under the curve (AUC)** : The ROC curve is a plot comparing true positive and false positive rates. The AUC metric is the area under the ROC curve. A model whose predictions are 100% wrong has an AUC of 0.0, one whose predictions are 100% right has an AUC of 1.0.
- **Mean Absolute Error (MAE):** Applicable for regression problems. The average difference between your model's predictions and the actual numbers.

- **Root Mean Squared Error (RMSE):** The square root of the average of squared differences between your model's predictions and the actual numbers.

2.3.4 Features

What features does your data have and which can you use to build your model? Not all data is the same. And when you hear someone referring to features, they're referring to different kinds of data within data. The three main types of features are categorical, continuous (or numerical), and derived.

- **Categorical Features:** One or the other(s). For example, in our heart disease problem, the sex of the patient. Or for an online store, whether or not someone has made a purchase or not.
- **Continuous or Numerical Features:** A numerical value such as average heart rate or the number of times logged in.
- **Derived Features:** Features you create from the data. Often referred to as feature engineering. Feature engineering is how a subject matter expert takes their knowledge and encodes it into the data. You might combine the number of times logged in with timestamps to make a feature called time since the last login. Or turn dates from numbers into "is a weekday (yes)" and "is a weekday (no)".

2.3.5 Modeling

Modeling breaks into three parts, choosing a model, improving a model, comparing it with others. When choosing a model, you'll want to take into consideration, interpretability, and ease to debug, amount of data, training, and prediction limitations.

2.3.5.1 Choosing a Model

- **Interpretability and easy to debug**: Ways that can reduce the errors. What is the internal working that is taking place?
- **Amount of Data:** How much data is required for the model to predict efficiently?
- **Training and Predicting Limitations:** This ties in with the above, how much time and resources do you have for training and prediction?

Linear models such as logistic regression are usually easier to interpret, are very fast for training, and predict faster than deeper models such as neural networks. But it's likely your data is from the real world. Data from the real world isn't always linear.

2.3.5.2 Tuning and Improving a Model

A model's first results aren't it's last. Like tuning a car, machine learning models can be tuned to improve performance. Tuning a model involves changing hyperparameters such as learning rate or optimizer. Or model-specific architecture factors such as the number of trees for random forests and number of and type of layers for neural networks.

2.3.5.3 Comparing Models

It's like comparing two apples based on their quality. Each model is trained on X and evaluated on Y but predictions vary.

2.3.6 Experimentation

This step involves all the other steps. Because machine learning is a highly iterative process, you'll want to make sure your experiments are actionable. Your biggest goal should be minimizing the time between offline experiments and online experiments. Offline experiments are the steps you take when your project isn't customer-facing yet. Online experiments happen when your machine learning model is in production.

All experiments should be conducted on different portions of your data.

- **Training Data Set:** Use this set for model training, 70–80% of your data is the standard.
- **Validation/Development Data Set:** Use this set for model tuning, 10–15% of your data is the standard.
- **Test Data Set:** Use this set for model testing and comparison, 10–15% of your data is the standard.

These amounts can fluctuate slightly, depending on your problem and the data you have. For example, consider a dataset with a million datasets. Then, the training data set constitutes 99.98% of the data and the validation and test set can have 0.01% data set each.

Poor performance on training data means the model hasn't learned properly. Try a different model, improve the existing one, collect more data, collect better data. Poor performance on test data means your model doesn't generalize well. Your model may be overfitting the training data. Use a simpler model or collect more data.

Poor performance once deployed (in the real world) means there's a difference in what you trained and tested your model on and what is happening. Revisit step 1 & 2. Ensure your data matches up with the problem you're trying to solve.

2.4 Pandas Library

Pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language. A fast and efficient DataFrame object for data manipulation with integrated indexing;

- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent data alignment and integrated handling of missing data: gain automatic

  label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible reshaping and pivoting of data sets;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Columns can be inserted and deleted from data structures for size mutability;
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;
- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- Time series-functionality: date range generation and frequency conversion, moving window statistics, date shifting, and lagging. Even createdomain-specific time offsets and join time series without losing data;
- Highly optimized for performance, with critical code paths written in Cython or C.
- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

2.5 Sci-Kit Learn Library

The Scikit-learn Python library, initially released in 2007, is commonly used in solving machine learning and data science problems—from the beginning to the end. The versatile library offers an uncluttered, consistent, and efficient API and thorough online documentation. Scikit-learn is an open-source Python library that has powerful tools for data analysis and data mining. It's available under the BSD license and is built on the following machine learning libraries:

- **NumPy:** Library for manipulating multi-dimensional arrays and matrices. It also has an extensive compilation of mathematical functions for performing various mathematical functions.
- **SciPy:** An ecosystem consisting of various libraries for completing technical

  computing tasks.
- **Matplotlib:** A library for plotting various charts and graphs.

Scikit-learn offers an extensive range of built-in algorithms that make the most of data science projects.
expression. We email text message, tweet, and update our statuses daily. As a result, unstructured text data has become extremely common, and analyzing large quantities of text data is now a key way to understand what people are thinking.

Chapter-3: Implementation

3.1 Code Structure

- The project is implemented by first importing the required libraries and our data into the jupyter notebook.
- We define various functions throughout our code to implement various library

  methods to achieve our objective.

3.2 Implementation Blocks

The objective of this part is to implement all the details we studied in the project designing and the machine learning pipeline part of the report. The implementation of all function blocks has been explained below.

3.2.1 Importing and Data Reading

- We import libraries: Scikit Learn, NumPy, Matplotlib and Pandas.
- We import the data we collected in an excel sheet into our dataframe using pandas.

3.2.2 Analysing dataset:

- We analyse the dataset collected and deduce which type of data is provided and how the data is structured.
- We analyse the quantity of the data along with the type of features, determining whether any value is missing or not.

3.2.3 Pre-processing the dataset:

- This step involves various steps in cleaning the dataset and removing any irrelevant feature present in the dataset.

- This step involves steps like dropping irrelevant features, filling any missing values, dealing with categorical features.

3.2.4 Feature correlation:

- In this step we determine the correlation between various features present in the dataset.
- For a model to work fine we need features that are highly correlated with the target features but the features should not be correlated with each other as this leads to bias result.

3.2.5 Exploratory data analysis:

- The selected features are explored in a graphical way.
- This step is important because it determines the relation between the target features and input features.

3.2.6 Splitting dataset in training and test data:

- For supervised machine learning it is very important to separate the test dataset from the machine until the model is completely ready for testing.
- The dataset splits in a ratio of 8:2 for training and testing.

3.2.7 Model selection:

- We import Logistic Regression function from scikit learn linear models.
- We import Decision Tree Classifier from scikit learn tree.
- We import Random Forest Classifier from scikit learn ensemble.

3.2.8 Training model:

- The selected model is trained with the training dataset.
- R-square score is recorded of the model to determine how well the model fitted our training data.

3.2.9 Testing model:

- We test the model with testing data using .predict method of the model.
- A new dataframe is created consisting of the actual value of the testing dataset and the result predicted by our model.

3.2.10 Analysing the performance of the model:

- We import confusion matrix, accuracy, recall, precision from sklearn.metrics
- We analyse the confusion matrix of the model.
- We record and analyse the various performance metrics of the model like accuracy, recall and precision.

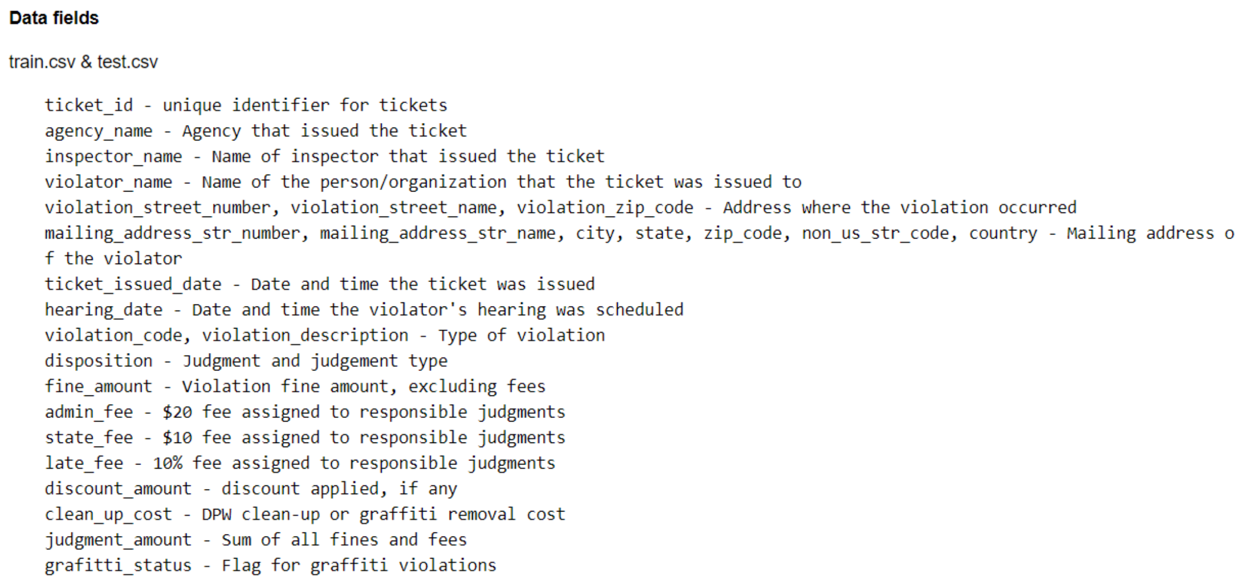
CHAPTER 4: RESULT AND DISCUSSION

About the dataset:

4.1 Result:

All the modules have been successfully implemented and further details have been discussed below.

4.1.1 Description about the Data Column



Data fields

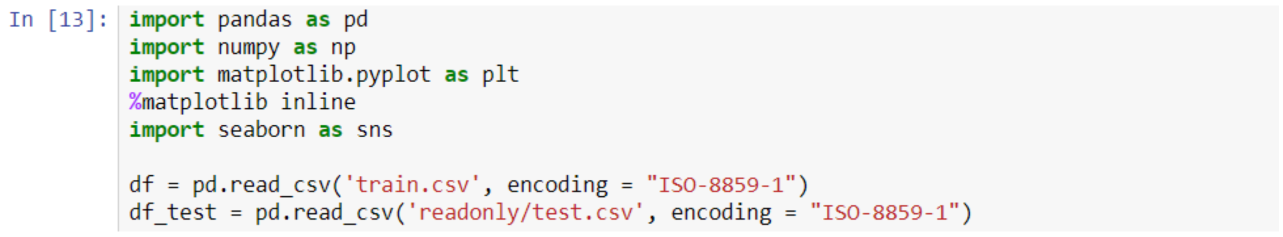
train.csv & test.csv

```
ticket_id - unique identifier for tickets
agency_name - Agency that issued the ticket
inspector_name - Name of inspector that issued the ticket
violator_name - Name of the person/organization that the ticket was issued to
violation_street_number, violation_street_name, violation_zip_code - Address where the violation occurred
mailing_address_str_number, mailing_address_str_name, city, state, zip_code, non_us_str_code, country - Mailing address o
f the violator
ticket_issued_date - Date and time the ticket was issued
hearing_date - Date and time the violator's hearing was scheduled
violation_code, violation_description - Type of violation
disposition - Judgment and judgement type
fine_amount - Violation fine amount, excluding fees
admin_fee - $20 fee assigned to responsible judgments
state_fee - $10 fee assigned to responsible judgments
late_fee - 10% fee assigned to responsible judgments
discount_amount - discount applied, if any
clean_up_cost - DPW clean-up or graffiti removal cost
judgment_amount - Sum of all fines and fees
grafitti_status - Flag for graffiti violations
```

The target variable is **compliance**, which is **True** if the ticket was paid **early, on time, or within one month** of the hearing data, **False** if the ticket was paid **after the hearing date or not at all, and Null** if the violator was found not responsible. Compliance, as well as a handful of other variables are only included in train.csv.

4.1.2 Importing and Reading Block:

The screenshot below shows the block where we import the basic libraries and also reads the data as a dataframe.

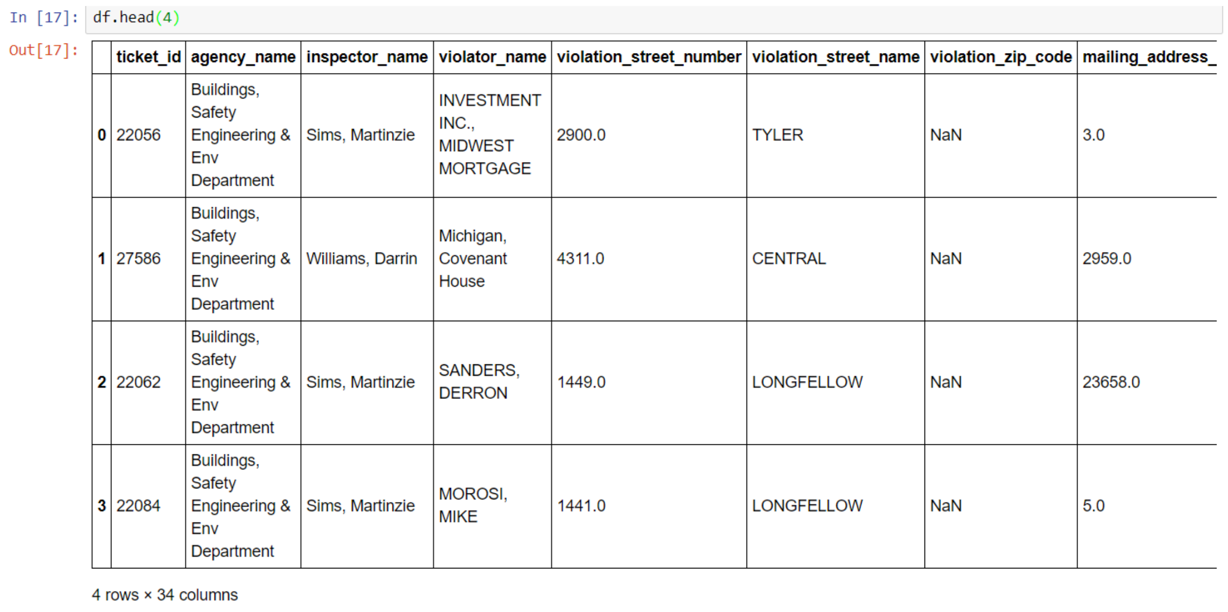


```
In [13]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline
         import seaborn as sns

         df = pd.read_csv('train.csv', encoding = "ISO-8859-1")
         df_test = pd.read_csv('readonly/test.csv', encoding = "ISO-8859-1")
```

Fig: Importing all the necessary libraries and converting data from CSV file to dataframe

4.1.3 Exploratory Data Analysis:

The screenshots below show the analysis step of the dataset where the dataset is understood that what types of data are provided and are there any missing values or not.

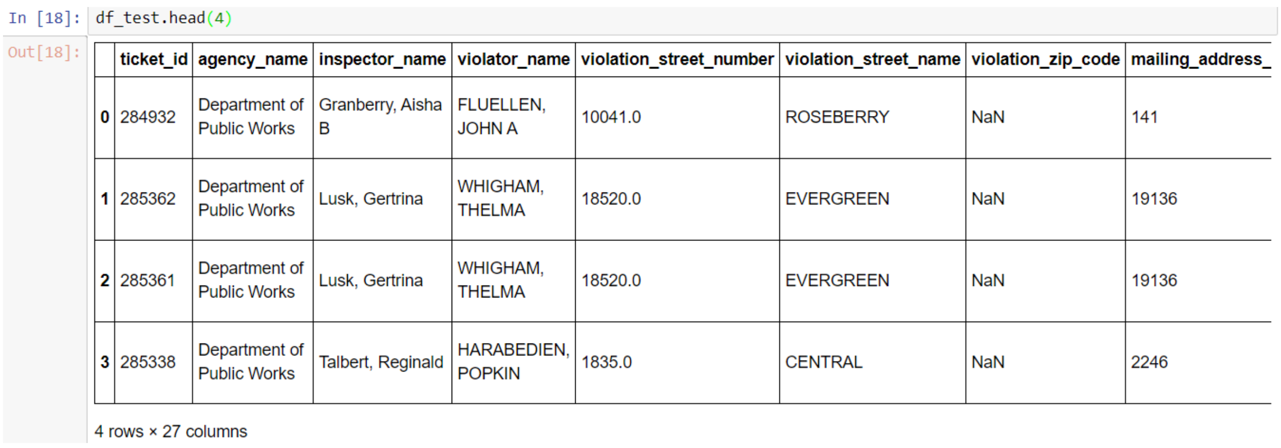


```
In [17]: df.head(4)
Out[17]:
```

	ticket_id	agency_name	inspector_name	violator_name	violation_street_number	violation_street_name	violation_zip_code	mailing_address_
0	22056	Buildings, Safety Engineering & Env Department	Sims, Martinzie	INVESTMENT INC., MIDWEST MORTGAGE	2900.0	TYLER	NaN	3.0
1	27586	Buildings, Safety Engineering & Env Department	Williams, Darrin	Michigan, Covenant House	4311.0	CENTRAL	NaN	2959.0
2	22062	Buildings, Safety Engineering & Env Department	Sims, Martinzie	SANDERS, DERRON	1449.0	LONGFELLOW	NaN	23658.0
3	22084	Buildings, Safety Engineering & Env Department	Sims, Martinzie	MOROSI, MIKE	1441.0	LONGFELLOW	NaN	5.0

4 rows × 34 columns

Fig: Pandas dataframe of the train dataset



```
In [18]: df_test.head(4)
Out[18]:
```

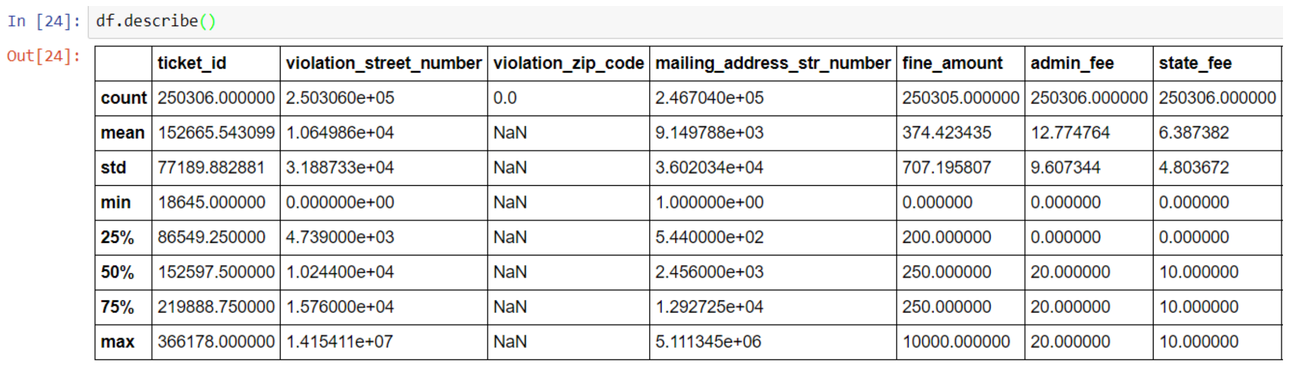
	ticket_id	agency_name	inspector_name	violator_name	violation_street_number	violation_street_name	violation_zip_code	mailing_address_
0	284932	Department of Public Works	Granberry, Aisha B	FLUELLEN, JOHN A	10041.0	ROSEBERRY	NaN	141
1	285362	Department of Public Works	Lusk, Gertrina	WHIGHAM, THELMA	18520.0	EVERGREEN	NaN	19136
2	285361	Department of Public Works	Lusk, Gertrina	WHIGHAM, THELMA	18520.0	EVERGREEN	NaN	19136
3	285338	Department of Public Works	Talbert, Reginald	HARABEDIEN, POPKIN	1835.0	CENTRAL	NaN	2246

4 rows × 27 columns

Fig: Pandas dataframe of the test dataset

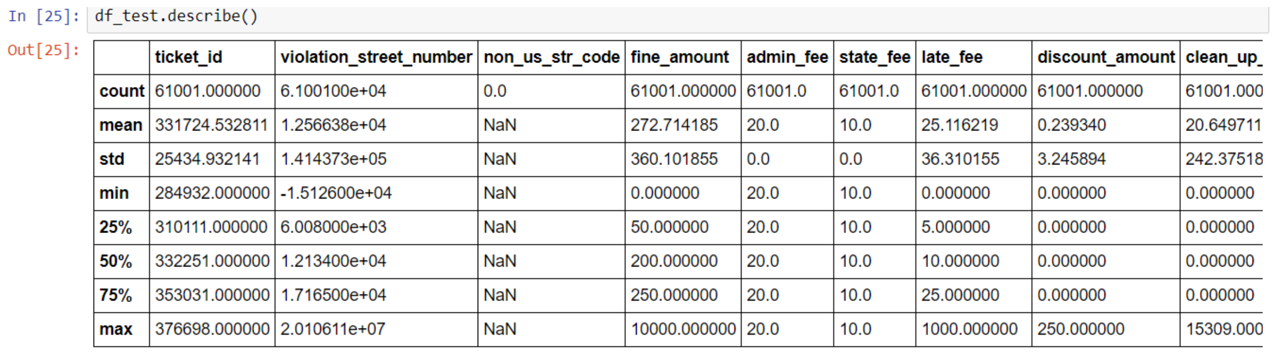
```
In [20]: df.shape

Out[20]: (250306, 34)

In [21]: df_test.shape

Out[21]: (61001, 27)

In [22]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250306 entries, 0 to 250305
Data columns (total 34 columns):
ticket_id                     250306 non-null int64
agency_name                   250306 non-null object
inspector_name                250306 non-null object
violator_name                 250272 non-null object
violation_street_number       250306 non-null float64
violation_street_name         250306 non-null object
violation_zip_code            0 non-null float64
mailing_address_str_number    246704 non-null float64
mailing_address_str_name      250302 non-null object
city                          250306 non-null object
state                         250213 non-null object
zip_code                      250305 non-null object
non_us_str_code               3 non-null object
country                       250306 non-null object
ticket_issued_date            250306 non-null object
hearing_date                  237815 non-null object
violation_code                250306 non-null object
violation_description         250306 non-null object
disposition                   250306 non-null object
fine_amount                   250305 non-null float64
admin_fee                     250306 non-null float64
state_fee                     250306 non-null float64
late_fee                      250306 non-null float64
discount_amount               250306 non-null float64
clean_up_cost                 250306 non-null float64
judgment_amount               250306 non-null float64
payment_amount                250306 non-null float64
```

```
In [23]: df_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61001 entries, 0 to 61000
Data columns (total 27 columns):
ticket_id                     61001 non-null int64
agency_name                   61001 non-null object
inspector_name                61001 non-null object
violator_name                 60973 non-null object
violation_street_number       61001 non-null float64
violation_street_name         61001 non-null object
violation_zip_code            24024 non-null object
mailing_address_str_number    59987 non-null object
mailing_address_str_name      60998 non-null object
city                          61000 non-null object
state                         60670 non-null object
zip_code                      60998 non-null object
non_us_str_code               0 non-null float64
country                       61001 non-null object
ticket_issued_date            61001 non-null object
hearing_date                  58804 non-null object
violation_code                61001 non-null object
violation_description         61001 non-null object
disposition                   61001 non-null object
fine_amount                   61001 non-null float64
admin_fee                     61001 non-null float64
state_fee                     61001 non-null float64
late_fee                      61001 non-null float64
discount_amount               61001 non-null float64
clean_up_cost                 61001 non-null float64
judgment_amount               61001 non-null float64
grafitti_status               2221 non-null object
dtypes: float64(9), int64(1), object(17)
```



```
In [24]: df.describe()
Out[24]:
```

	ticket_id	violation_street_number	violation_zip_code	mailing_address_str_number	fine_amount	admin_fee	state_fee
count	250306.000000	2.503060e+05	0.0	2.467040e+05	250305.000000	250306.000000	250306.000000
mean	152665.543099	1.064986e+04	NaN	9.149788e+03	374.423435	12.774764	6.387382
std	77189.882881	3.188733e+04	NaN	3.602034e+04	707.195807	9.607344	4.803672
min	18645.000000	0.000000e+00	NaN	1.000000e+00	0.000000	0.000000	0.000000
25%	86549.250000	4.739000e+03	NaN	5.440000e+02	200.000000	0.000000	0.000000
50%	152597.500000	1.024400e+04	NaN	2.456000e+03	250.000000	20.000000	10.000000
75%	219888.750000	1.576000e+04	NaN	1.292725e+04	250.000000	20.000000	10.000000
max	366178.000000	1.415411e+07	NaN	5.111345e+06	10000.000000	20.000000	10.000000

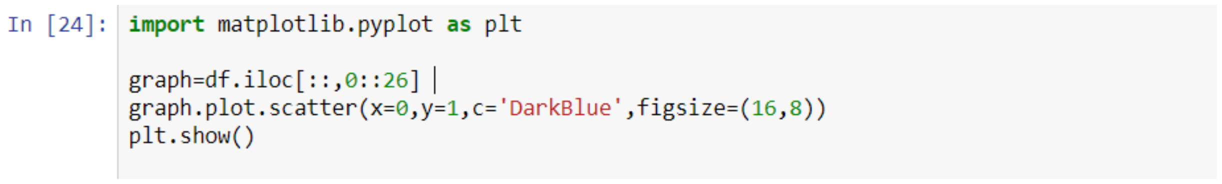


```
In [25]: df_test.describe()
Out[25]:
```

	ticket_id	violation_street_number	non_us_str_code	fine_amount	admin_fee	state_fee	late_fee	discount_amount	clean_up_
count	61001.000000	6.100100e+04	0.0	61001.000000	61001.0	61001.0	61001.000000	61001.000000	61001.000
mean	331724.532811	1.256638e+04	NaN	272.714185	20.0	10.0	25.116219	0.239340	20.649711
std	25434.932141	1.414373e+05	NaN	360.101855	0.0	0.0	36.310155	3.245894	242.37518
min	284932.000000	-1.512600e+04	NaN	0.000000	20.0	10.0	0.000000	0.000000	0.000000
25%	310111.000000	6.008000e+03	NaN	50.000000	20.0	10.0	5.000000	0.000000	0.000000
50%	332251.000000	1.213400e+04	NaN	200.000000	20.0	10.0	10.000000	0.000000	0.000000
75%	353031.000000	1.716500e+04	NaN	250.000000	20.0	10.0	25.000000	0.000000	0.000000
max	376698.000000	2.010611e+07	NaN	10000.000000	20.0	10.0	1000.000000	250.000000	15309.000

Fig: Details about all the features

4.1.4 Scatter Plot for Categorical Features



```
In [24]: import matplotlib.pyplot as plt

         graph=df.iloc[::,0::26] |
         graph.plot.scatter(x=0,y=1,c='DarkBlue',figsize=(16,8))
         plt.show()
```

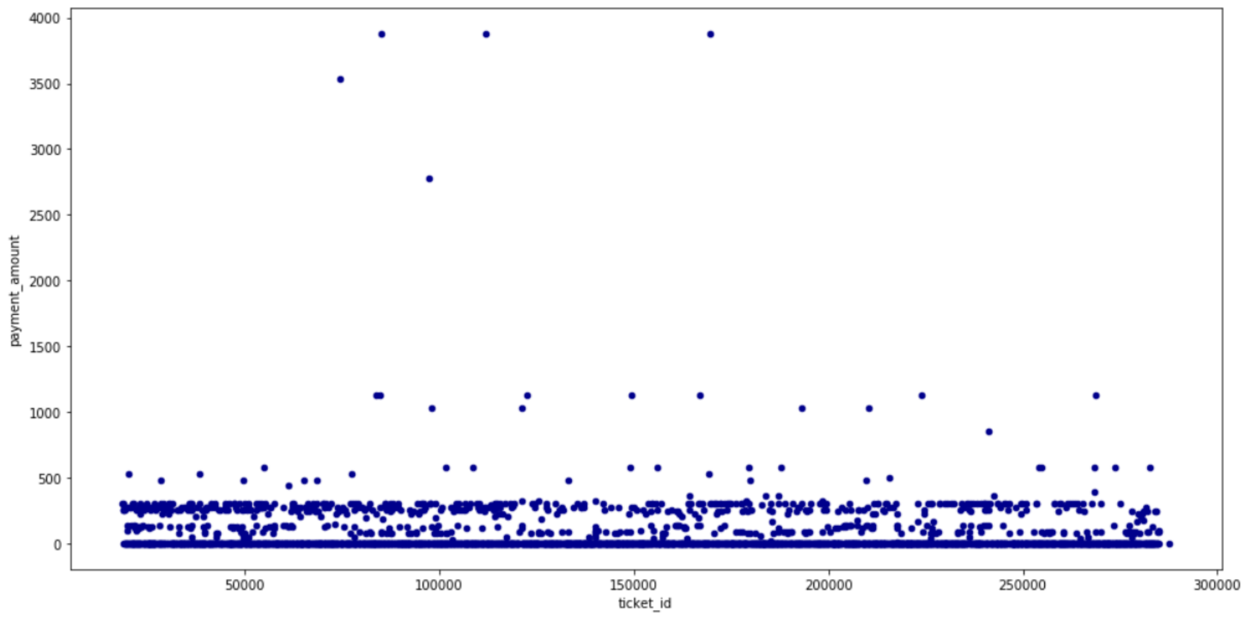


Fig: The plot confirms most of the fines are likely to be under 500.

```
In [35]: import matplotlib.pyplot as plt

         graph=df.iloc[::,0::33]
         graph.plot.scatter(x=0,y=1,c='DarkBlue',figsize=(16,8),s=1)
         plt.show()
```

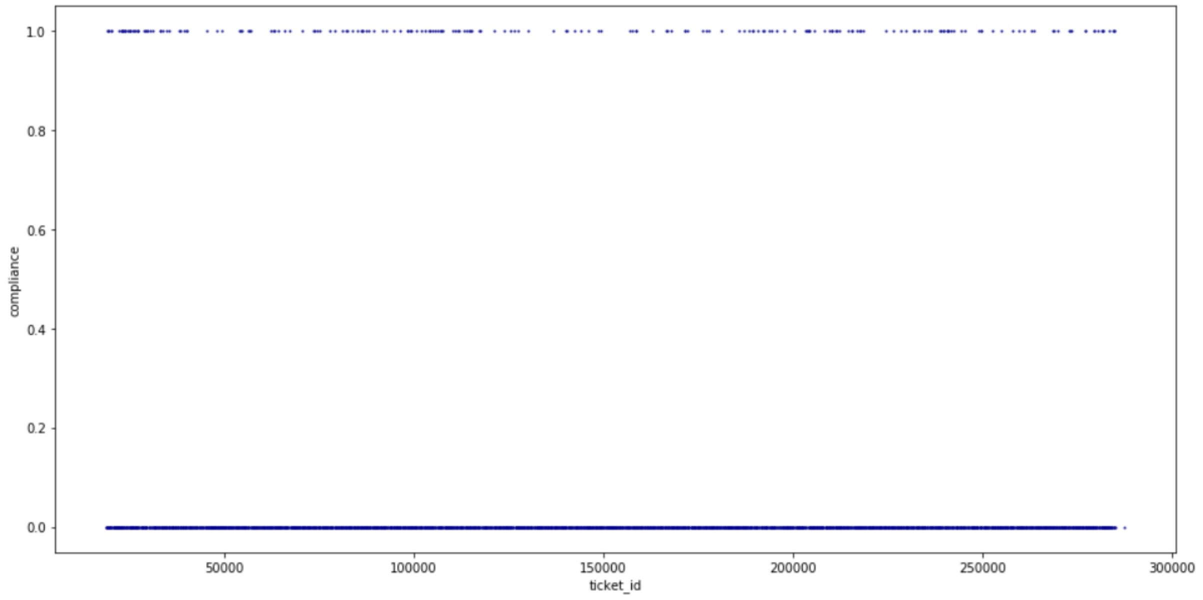


Fig: This plot confirms that the fines are most likely to be unpaid as the compliance is most likely to be 0.0

4.1.5 Modelling

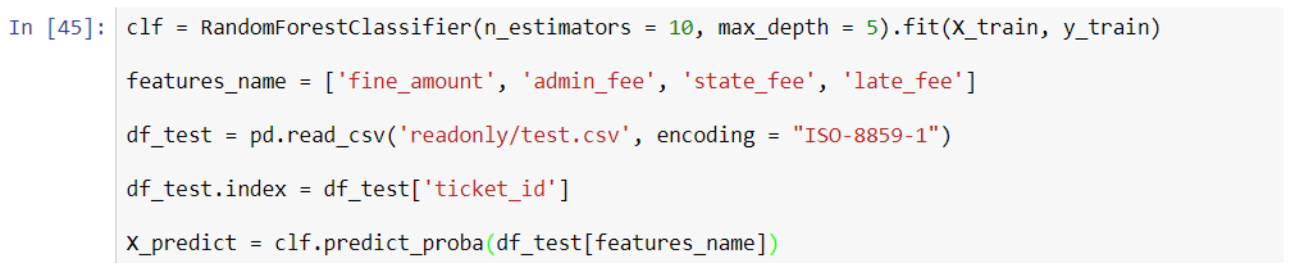
Importing Classifier Modules

```
In [38]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve, auc
         from sklearn.preprocessing import LabelEncoder
         from sklearn.model_selection import GridSearchCV
```

Splitting the data using train_test_split

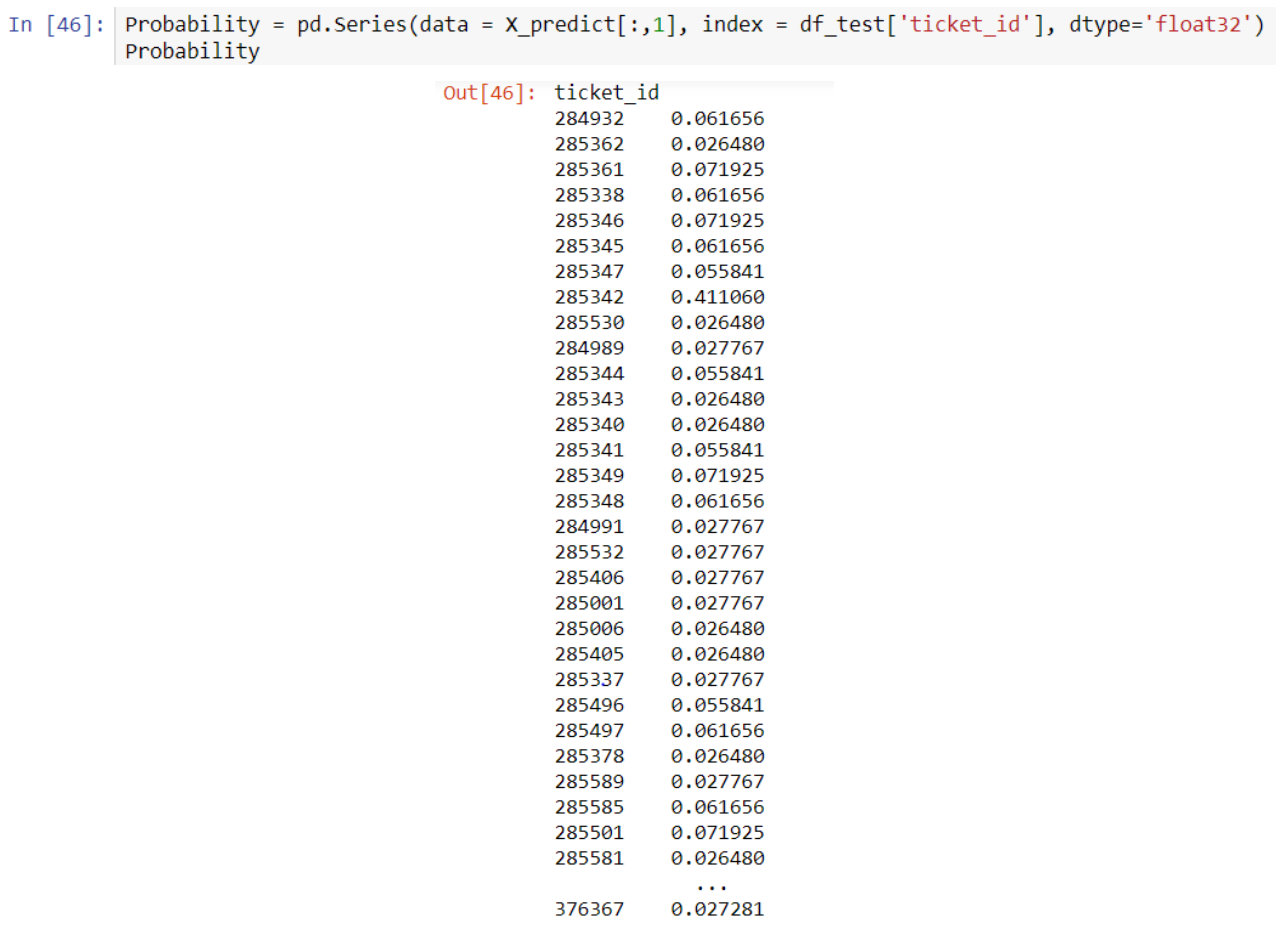
```
In [44]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

Random Forest Classifier



```
In [45]: clf = RandomForestClassifier(n_estimators = 10, max_depth = 5).fit(X_train, y_train)

         features_name = ['fine_amount', 'admin_fee', 'state_fee', 'late_fee']

         df_test = pd.read_csv('readonly/test.csv', encoding = "ISO-8859-1")

         df_test.index = df_test['ticket_id']

         X_predict = clf.predict_proba(df_test[features_name])
```

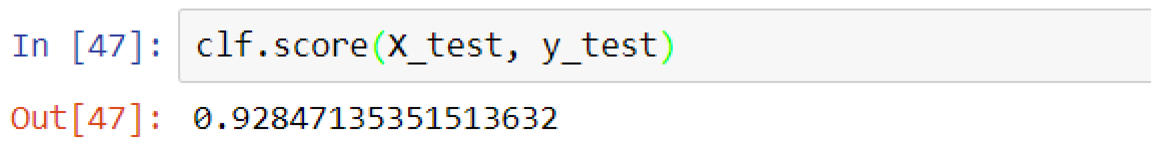
Fig: Building the model



```
In [46]: Probability = pd.Series(data = X_predict[:,1], index = df_test['ticket_id'], dtype='float32')
         Probability

Out[46]: ticket_id
         284932    0.061656
         285362    0.026480
         285361    0.071925
         285338    0.061656
         285346    0.071925
         285345    0.061656
         285347    0.055841
         285342    0.411060
         285530    0.026480
         284989    0.027767
         285344    0.055841
         285343    0.026480
         285340    0.026480
         285341    0.055841
         285349    0.071925
         285348    0.061656
         284991    0.027767
         285532    0.027767
         285406    0.027767
         285001    0.027767
         285006    0.026480
         285405    0.026480
         285337    0.027767
         285496    0.055841
         285497    0.061656
         285378    0.026480
         285589    0.027767
         285585    0.061656
         285501    0.071925
         285581    0.026480
                     ...
         376367    0.027281
```

Fig: Probability whether a given blight ticket will be paid on time

Random Forest Score



```
In [47]: clf.score(X_test, y_test)

Out[47]: 0.92847135351513632
```

4.2 Discussions

4.2.1 Important metric to consider for model performance:

Out of all the performance metrics available, it is important to pay more attention to recall because as per our problem, we need a model which has low variance and low bias.

4.2.2 Why Random Forest?

Whether you have a regression or classification task, random forest is an applicable model for your needs. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed. Random forests are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. They also provide two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy.

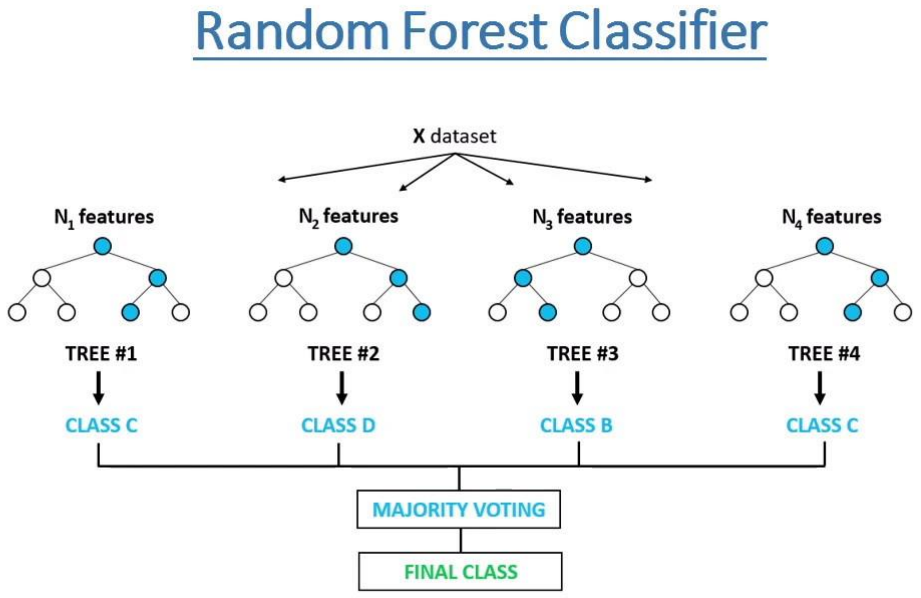


Fig: Random Forest Classifier flow chart

Advantages of Random Forest

i. Random Forest is a powerful algorithm based on the **bagging** algorithm and uses **Ensemble Learning** technique. It creates as many trees on the subset of the data and combines the output of all the trees. In this way it **reduces overfitting** problem in decision trees and also **reduces the variance** and therefore **improves the accuracy**.

ii. Random Forest can be used to **solve both classification as well as regression problems**.

iii. Random Forest works well with both **categorical and continuous variables**.

iv. Random Forest can automatically **handle missing values**.

v. **No feature scaling required:** No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule-based approach instead of distance calculation.

vi. **Handles non-linear parameters efficiently:** Non-linear parameters don't affect the performance of a Random Forest unlike curve-based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform as compared to other curve-based algorithms.

vii. Random Forest can automatically **handle missing values**.

viii. Random Forest is usually **robust to outliers** and can handle them automatically.

ix. Random Forest algorithm is very **stable**. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees.

x. Random Forest is comparatively **less impacted by noise.**

CHAPTER 5: FUTURE SCOPES AND CONCLUSIONS

5.1 Future scope:

Each module of the application has been written in a modular way, which makes it easy to upgrade.

The following upgrades could be made in the future:

i. With changing data of the citizens, we can make the model in such a way that it increases the train dataset which in turn increases the predicting power of our model.

ii. We can do hyper-parameter tuning of our model to increase its predicting power.

iii. We can also deploy other models to see whether the results can be improved further.

5.2 Conclusions

All the objectives were completed in the given frame of time. It is also concluded that the Random Forest Classifier (RFC) provides the best recall score for this problem. We can although with the further tuning of hyperparameters may prefer some other model but this gives a rough idea that RFC provides the best results.

Task Accomplished

i. Successfully completed the analysis of people who are more likely to pay the blight fines in the City of Detroit.

ii. Analysed the data and features of the dataset.

iii. Explored the data by plotting scatter plots.

iv. Trained the model to accomplish the business problem.

v. Tested the final model on the test dataset and analysed the results of the model.

REFERENCES

Since data science is a collective work of studying, implementing, correcting and applying, the following sites and blogs, videos were used for research and study purpose throughout the project completion period to complete the project:

i. https://data.detroitmi.gov/
ii. https://data.detroitmi.gov/Property-Parcels/Building-Permits/xw2a-a7tf
iii. https://detroitmi.gov/How-Do-I/Report/Blight-Complaint-FAQs
iv. http://www.codeastar.com/data-wrangling/
v. https://scikit-learn.org/stable/
vi. https://stackoverflow.com/
vii. https://medium.com/@yrnigam/how-to-write-a-data-science-report-181bd49d8f4d
viii. https://pandas.pydata.org/docs/
ix. https://en.wikipedia.org/wiki/Data_science

Book referred:

i. Hands–On Machine Learning with Scikit–Learn and TensorFlow by Aurelien Geron.

APPENDICES

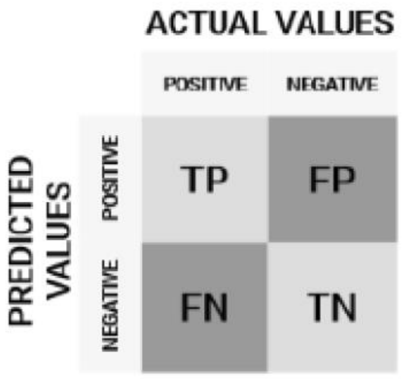


Fig: Confusion matrix

i. **Actual Positive:** When the model predicted positive and it is actually positive.

ii. **Actual Negative:** When the model predicted negative and it is actually negative.

iii. **False Positive:** When the model predicted positive but it is actually negative.

iv. **False Negative:** When the model predicted negative but it is actually positive.

v. **Accuracy:** Accuracy is the number of correctly predicted data points out of all the data points.

vi. **Precision** = TruePositives / (TruePositives + FalsePositives)

vii. **Recall** = TruePositives / (TruePositives + FalseNegatives)