

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering Degree in Computer Science and Engineering

By

SIVA RAMA KRISHNA.M
(Reg.No. 40731057)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

OCT 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Siva Rama Krishna.M** who carried out the project entitled “**USING DATASET TO PREDICT AUTO MPG**” under my supervision from Aug 2022 to Oct 2022.

Internal Guide

Dr.J.Albert Mayan M.E., Ph.D.

Head of the Department

Dr.L.LAKSHMANAN M.E., Ph.D.

Submitted for Viva voce Examination held on 09/11/2022

Internal Examiner

External Examiner

DECLARATION

I, **Siva Rama Krishna.M** (Reg No: **40731057**) hereby declare that the Project Report entitled **“USING DATASET TO PREDICT AUTO MPG”** Done by me under the guidance of **Dr.J.Albert Mayan M.E,Phd.** and **Mr.Hari at Cognibot ML** submitted in Partial fulfilment of the requirements for the award of Bachelore of Engineering degree In Computer Science and Engineering.

DATE: 09/11/2022

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr.L.Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.J.Albert Mayan M.E., Ph.D.** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my projectwork.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

In this project, we mainly discussed about fuel efficiency in the cars. We only used the Auto-MPG dataset which includes nine basic attributes of the cars. In order to understand the correlation between the attributes of the cars, we make the dataset suitable for the functions that we will use. Thereafter we did visualization, some subtasks and predict the mpg with using Both Random Forest and Decision Tree method. The fuel of the automobile like cars mainly states the distance which is travelled by a car and particular fuel consumption is consumed by car or any vehicle. As we know consumption of fuel through vehicles like car play major role in air pollution and consumption of fuel varies from one country to another country. Improving the fuel consumptions according to factors that influence. We are witnessing the fuel prices and customer being more particular about the features. To have a vehicle which is desirable and even more efficient, improvement of fuel is been carried out. Mostly, fuel price varies from country to country and also depends upon the car model those countries use thirdly, how customers often uses the car. Like daily bases, only for vacation purpose as this plays a major role. So, many factors come into picture which have more influence in affecting fuel consumption to draw a conclusion. That also affects increment in the price of fuel or decrement in the price of fuel. Considering all the factors which affects fuel. By understanding each and every factor which influence MPG (miles per gallon) that is fuel and analyzing those factors.

KEYWORDS: Data analyzing, Data Pre-processing, Decision Trees And Random Forest

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
2.1	Excel Sheet of the Given Auto-MPG Dataset	4
2.2	Missing Attributes	4
4.1	Training and Testing the Data	15

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	Libraries of Python	6
3.2	Matplotlib	7
3.3	Example For Decision Tree	9
3.4	Random Forest Outlier	10
4.1	Flow Chart For Random Forest	12
4.2	Decision Tree Regression Model	13
4.3	Decision Tree Regression Model V2	13

TABLE OF CONTEXTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
1	INTRODUCTION	1
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	2
	2.1 Aim	2
	2.2 Scope	2
	2.3 Attribute Information	3
	2.4 Dataset Description	3
	2.5 Missing Attributes:(denoted by “null”)	4
3	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	5
	3.1 Data Preprocessing	5
	3.2 Import Necessary Libraries	5
	3.3 Algorithms Used	8
4	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	11
	4.1 Result:	11
	4.2 Discussion About Random Forest:	11
	4.3 Discussion About Decision Tree:	12
	4.4 Strength and Weakness of Each Model:	12
	4.5 Training ,Testing and Finding Accuracy (Model Analysis)	14
5	SUMMARY AND CONCLUSIONS	16
	5.1 Reference	16
	5.2 Appendix	17
	A. Screenshots:	17-23
	B. Source Code:	24-29

CHAPTER 1

INTRODUCTION

Miles per gallon (MPG) is used to measure a fuel in vehicles like cars. It can be also defined as the distance travelled by vehicle. Based on origins MPG have different values. MPG of vehicle can be viewed through graphs. Through many factors like displacement, horsepower, weight, car name, acceleration, model. Also, focusing on the terms which produced to predict our MPG values. So, by this we can develop and update on cars functions. As in the dataset we have following attributes Displacement is one factor tells us about the sucked air by engine pistons. Cylinder is another factor gives information about the gasoline is burned and turned into power. Weight describes about the Particular automobile in pounds or kilograms. Acceleration which tells us about magnitude and direction. Horsepower mainly gives us the information about rate at which work done by the engine. Model tells us about year of production of car. Developed model on this dataset will be trained on many different cars that too different models of that car as we know role of these factors play a major role on different model of cars. Training model with proper information, predicting fuel through various factors and it should also give us a good estimate for unknown car's MPG. By using optimization techniques, we can reduce error and increase the performance of the model.

Understanding the factors that ability to influence consumption and predicting them, is vital in knowing vehicle fuels. For example, if we consider a car travelling from one place to another place where the place is like hilly road. So, based on the car model, Number of cylinders does car have and based on the weight, displacement, etc under certain features our fuel is consumed. So, mostly on hilly areas and also due to traffic, speed breakers fuel consumption will be more. This fuel consumption of automobile(car) varies from country to country. It also depends on model of the car. Mostly customers observe the various factors to buy a car mainly they prefer low consumption fuel automobiles(car). It also depends how customers will use these cars. Training model with proper information, predicting fuel through various factors and it should also give us a good estimate for unknown car's MPG. By using optimization techniques, we can reduce error and increase the performance of the model.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

To predict the values of Auto-MPG by Using the dataset "auto-mpg.csv" we will try to create a model to predict car consumption (MPG or miles per gallon).

2.2 SCOPE

The automotive industry is extremely competitive. With increasing fuel prices and picky consumers, automobile makers are constantly optimizing their processes to increase fuel efficiency.

But, what if you could have a reliable estimator for a car's mpg given some known specifications about the vehicle?

Then, you could beat a competitor to market by both having a more desirable vehicle that is also more efficient, reducing wasted R&D costs and gaining large chunks of the market.

According to others using this dataset, some of the mpg values for the cars are incorrect, meaning that some of our predictions will be off by a large amount, but we shouldn't always trust the listed mpg value.

There are also unknown mpg values in the dataset, marked with a '?'. We will need to manually replace these with the correct mpg value.

While our model is the end result of this notebook, the data analysis section will be incredibly important in visualizing trends without having to use any machine learning techniques.

2.3 ATTRIBUTE INFORMATION

- 1.mpg: continuous
- 2.cylinders: multi-valued discrete
- 3.displacement: continuous
- 4.horsepower: continuous
- 5.weight: continuous
- 6.acceleration: continuous
- 7.model year: multi-valued discrete
- 8.origin: multi-valued discrete
- 9.car name: string (unique for each instance)

2.4 DATASET DESCRIPTION

Dataset mainly contains 9 attributes with 398 rows. MPG is a target variable it is a numerical data ranges from [9- 46.6] which explains about the millage per gallon (MPG). It is measurement of a fuel consumption of an automobile car in which we need to predict MPG with various factors like Displacement, Cylinder, Acceleration, Horsepower, Weight, Model, Origin, Car name of three countries like America, European, Asia. In this dataset, America is mapped as origin1, Asia as origin2 and Europe as origin3 which describes Origin attribute. Displacement attribute is a numerical data which ranges from [68-455] tells us about sucked air by engine pistons. Cylinder attribute is a numerical data which ranges from [3- 8] gives information about the gasoline is burned and turned into power. Weight attribute is also a numerical data which ranges from [1613-5140] describes about weight of particular automobile in pounds or kilograms. Acceleration attribute is also a numerical float data which ranges from [8-24.8] which tells us about magnitude and direction. Horsepower attribute is numerical which ranges from [46-230] but it includes few missing values. It mainly gives us the information about rate at which

work done by the engine. Model attribute tells us about the year of production of car it ranges from [70- 82]. Car name attribute is the categorical data which gives the name of a car.

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11	70	1	plymouth satellite
16	8	304	150	3433	12	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
15	8	429	198	4341	10	70	1	ford galaxie 500
14	8	454	220	4354	9	70	1	chevrolet impala
14	8	440	215	4312	8.5	70	1	plymouth fury iii
14	8	455	225	4425	10	70	1	pontiac catalina
15	8	390	190	3850	8.5	70	1	amc ambassador dpl
15	8	383	170	3563	10	70	1	dodge challenger se
14	8	340	160	3609	8	70	1	plymouth cuda 340
15	8	400	150	3761	9.5	70	1	chevrolet monte carlo
14	8	455	225	3086	10	70	1	buick estate wagon (sw)
24	4	113	95	2372	15	70	3	toyota corona mark ii
22	6	198	95	2833	15.5	70	1	plymouth duster
18	6	199	97	2774	15.5	70	1	amc hornet
21	6	200	85	2587	16	70	1	ford maverick
27	4	97	88	2130	14.5	70	3	datson p1510
26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan
25	4	110	87	2672	17.5	70	2	peugeot 504
24	4	107	90	2430	14.5	70	2	audi 100 ls
25	4	104	95	2375	17.5	70	2	saab 99e
26	4	121	113	2234	12.5	70	2	bmw 2002
21	6	199	90	2648	15	70	1	amc gremlin
10	8	360	215	4615	14	70	1	ford f250

TAB 2.1 – Excel Sheet of the Given Auto-MPG Dataset

2.5 MISSING ATTRIBUTES: (denoted by “null”)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   mpg              398 non-null    float64
1   cylinders        398 non-null    int64
2   displacement     398 non-null    float64
3   horsepower       398 non-null    object
4   weight           398 non-null    int64
5   acceleration     398 non-null    float64
6   model year      398 non-null    int64
7   origin           398 non-null    int64
8   car name        398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

TAB 2.2 – Missing Attributes

CHAPTER 3

EXPERIMENTAL OR MATERIAL AND METHODS, ALGORITHMS USED

3.1 DATA PREPROCESSING

The purpose of the data preprocessing stage is to minimize potential errors in the model as much as possible.

Generally, a model is only as good as the data passed into it, and the data preprocessing we do ensures that the model has as accurate a dataset as possible.

While we cannot perfectly clean the dataset, we can at least follow some basics steps to ensure that our dataset has the best possible chance of generating a good model.

First, let's check and see null values for this dataset. Null values are empty, useless entries within our dataset that we don't need. If we skip removing null values, our model will be inaccurate as we create "connections" for useless values, rather than focusing all resources onto creating connections for useful values.

3.2 Import Necessary Libraries

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's I look at the libraries that came in handy for mammographic dataset.

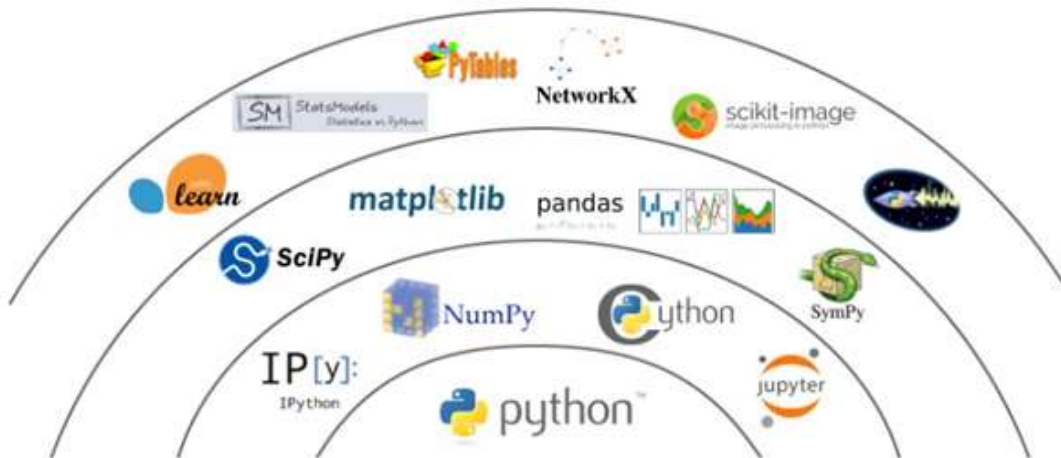


FIG 3.1 – Libraries of Python

It is important to import all the necessary libraries such as
pandas, NumPy, matplotlib, seaborn, Sklearn

- **Numpy** – Numpy stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of countless of routines for processing those arrays. Using this mathematical and logical operations on arrays can be performed. The difference in using Numpy from pandas is, it works on numerical data whereas pandas on tabular data.
- **Pandas** – Pandas is a widely-used data analysis and manipulation library for python. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. IT also provides fast, flexible and expressive data structures working with relational or labeled or both easy and intuitive. Considered as fundamental high-level building block in performing practical, real-world data analysis in python. Has powerful tools like DataFrame and Series for analyzing.
- **Seaborn** – Seaborn is a library built on top of matplotlib. It used for data visualization and exploratory data analysis. They work easily with dataframes and pandas library. The graphs created can also be customized easily. It provides default styles and color palettes to make statistical plots more attractive. Also closely integrated to the data structures from pandas

- **Matplotlib** – Visualisation with python. It is a comprehensive library for creating static, animated, and interactive visualizations in Python

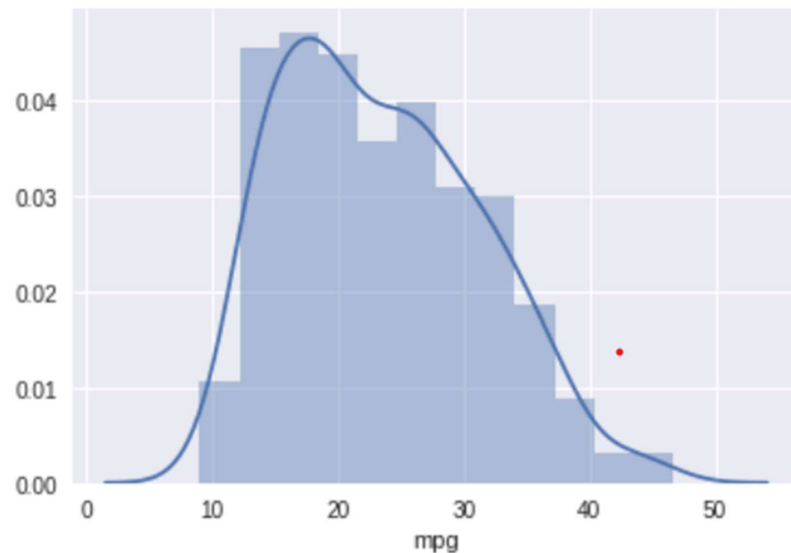


FIG 3.2 – MATPLOTLIB

- **Sklearn** – which contains all the modules required for model building. It stands for Scikit-learn, a machine learning library. It is imported for various classification, regression and clustering algorithms including k-means random forest, support vector machines, gradient boosting and DBSCAN. It is designed using libraries Numpy and Scipy. From the sklearn library and from the tree inside the library DecisionTreeClassifier. It is a class capable of performing multi-class classifier on a dataset. When compared with other classifiers, DecisionTreeClassifier takes input as two arrays: an array X, a sparse or dense, of shape (n_samples, n_features) holding training samples and an array Y of integer values, shape (n_samples), holding class labels for training sample. From sklearn another one called model_selection for training and testing the model imports train_test_split. It is a method setting a blueprint to analyze data and the using it to measure new data. Selecting a proper model allows to generate accurate results while making prediction. For proceeding, we need to train the model by using a specific dataset and test the model against another dataset. By default, sklearn train_test_split will make random partitions for two subsets. We can also specify a random

state for the operation. First, we need to split the dataset and then allocate the size for train and test. For this mammographic dataset we need train size as 70% (0.70) and test size as 30% (0.30) with the random state of 100.

- **Operator** module exports a set of efficient functions corresponding to the intrinsic operators of Python. For example, `operator.add(x, y)` is equivalent to the expression `x+y`. Many function names are those used for special methods, without the double underscores. For backward compatibility, many of these have a variant with the double underscores kept. The variants without the double underscores are preferred for clarity. The functions fall into categories that perform object comparisons, logical operations, mathematical operations and sequence operations.

3.3 Algorithms Used

Decision Tree:

Decision Trees build upon continuously to partition the data. The aim of decision tree is to increase the predictiveness as much as possible at each stage so that the model keeps gaining information about the dataset. Randomly splitting will not give us valuable insight into dataset. The purity of node is inversely proportional to distribution of different classes in that node. Overfitting model would be too specific model and not be generalize well. Though it achieves high accuracy with training set but poorly on new. The depth of the tree is controlled by `max_depth` parameter for decision tree algorithm in scikit-learn. Is also suitable to work on a mixture of feature data types.

Decision trees “utilize a tree structure to model the relationships among features and the potential outcomes... this structure earned its name due to the fact that it mirrors how a literal tree begins at a wide trunk, which if followed upward, splits into narrower and narrower branches. In much the same way, a decision tree classifier uses a structure of branching decisions which channel examples into a final predicted class value”

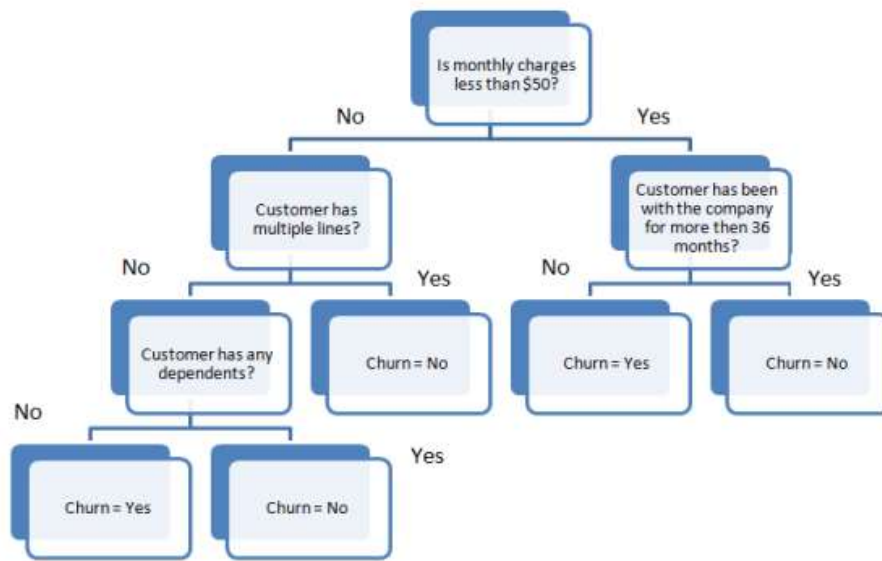


FIG 3.3 – Example For Decision Tree

A.Implementation of Decision Tree

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a regressor model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

Random Forest:

Random Forest is an ensemble of many decision trees. They are built using a method called bagging where decision trees are used as parallel estimators. When used in classification problem, the result will be based on majority of vote received from each decision tree. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model*.

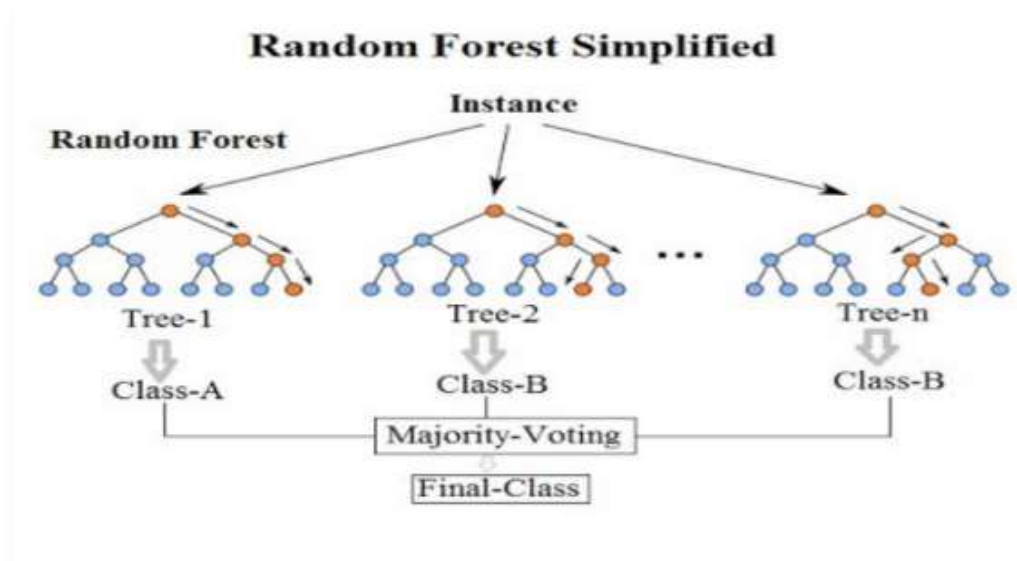


FIG 3.4 – Random Forest Outlier

CHAPTER 4

RESULTS AND PERFORMANCE ANALYSIS

4.1 RESULT:

After we have analyzed the data, we can see that there is a pattern with this data. Usually most of the cars have 4 cylinders and they are top tier with mpg. Weight of the car has the most impact on mpg. People tend to buy their cars from the brand Ford. The most efficient year was 1980 and after that with the data we have, we can see that it decreases next 2 years. Finally, with random forest and decision tree method we are able to predict mpg values.

4.2 DISCUSSION ABOUT RANDOM FOREST:

Regression is the other task performed by a random forest algorithm. A random forest regression follows the concept of simple regression. Values of dependent (features) and independent variables are passed in the random forest model.

We can run random forest regressions in various programs such as SAS, R, and python. In a random forest regression, each tree produces a specific prediction. The mean prediction of the individual trees is the output of the regression. This is contrary to random forest classification, whose output is determined by the mode of the decision trees' class.

Although random forest regression and linear regression follow the same concept, they differ in terms of functions. The function of linear regression is $y = bx + c$, where y is the dependent variable, x is the independent variable, b is the estimation parameter, and c is a constant. The function of a complex random forest regression is like a blackbox.

4.3 DISCUSSION ABOUT DECISION TREE:

Decision trees “utilize a tree structure to model the relationships among features and the potential outcomes... this structure earned its name due to the fact that it mirrors how a literal tree begins at a wide trunk, which if followed upward, splits into narrower and narrower branches. In much the same way, a decision tree classifier uses a structure of branching decisions which channel examples into a final predicted class value”

4.4 STRENGTH AND WEAKNESS OF EACH MODEL:

RANDOM FOREST

Random forest runtimes are quite fast, and they are able to deal with unbalanced and missing data. Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy. Of course, the best test of any algorithm is how well it works upon your own data set.

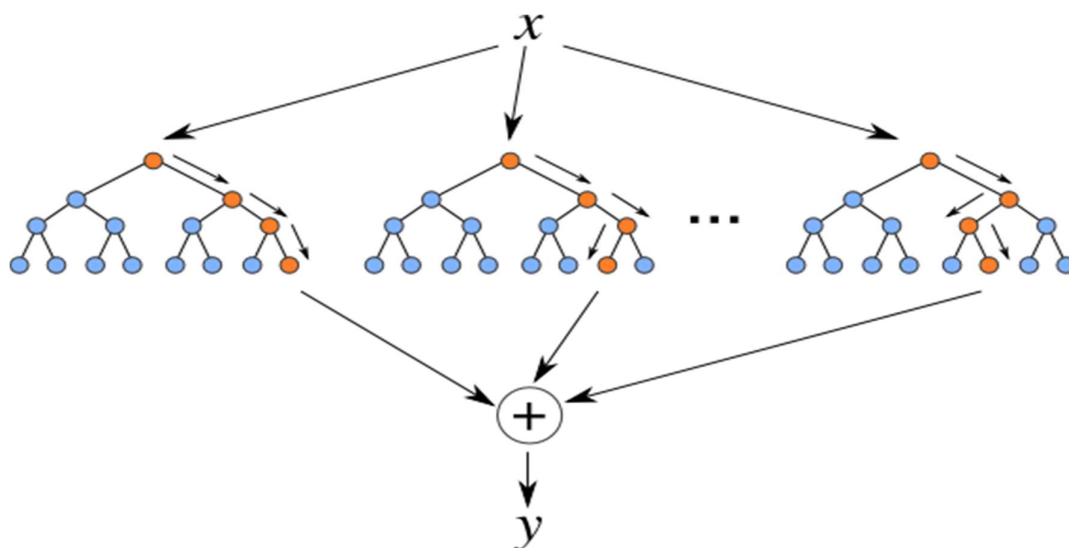


FIG 4.1 – Flow Chart For Random Forest

DECISION TREE

Decision trees are great at looking back at models. You can take prior data and create a well-made model that will capture all the data within beforehand that can easily provide solid results at point within the model. Where the model can fail, however, is when it is used to forecast future results.

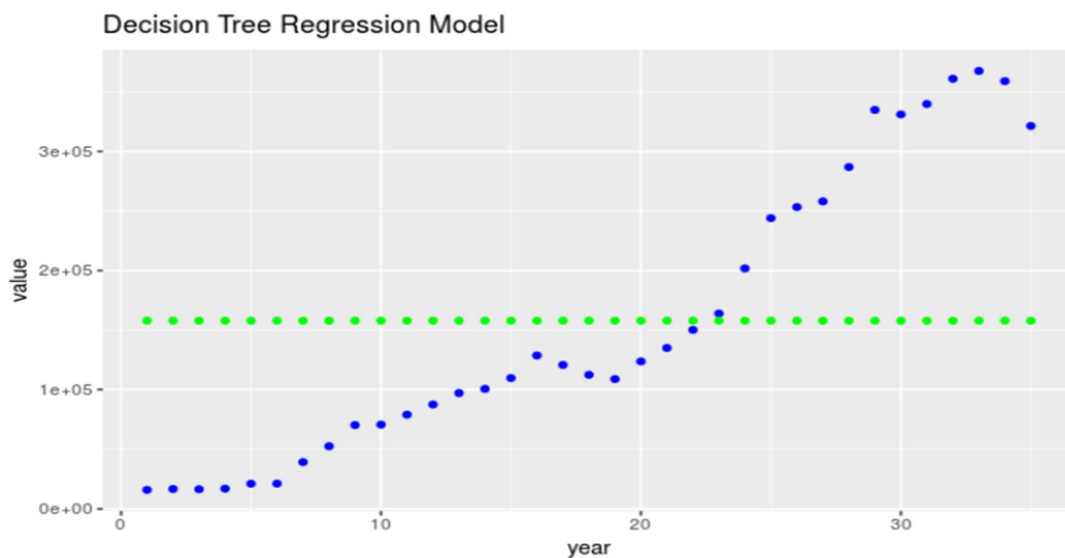


FIG 4.2 – Decision Tree Regression Model

Also, the models take some understanding as to how to tune them. The above graph captures the data only at one point - at about 23 years in, the model is perfect, but all other points are completely ignored. In this situation, you would be better off averaging all the data points to get your prediction.

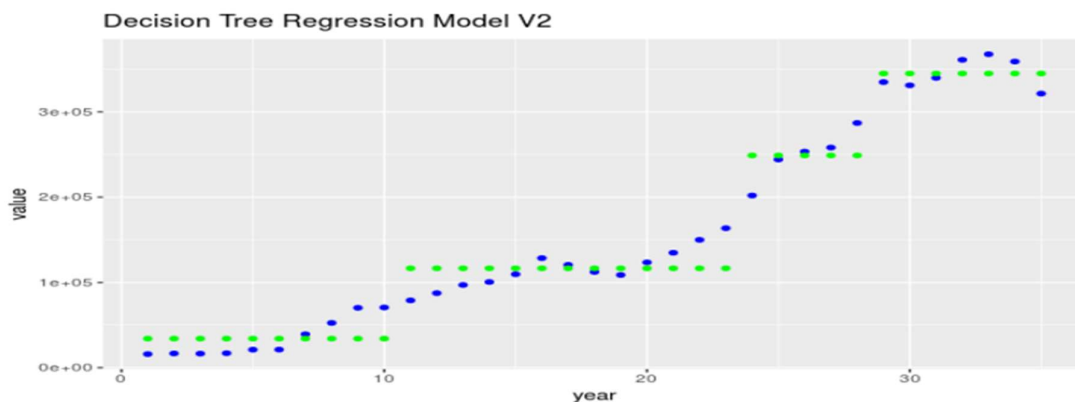


FIG 4.3 – Decision Tree Regression Model V2

4.5 TRAINING ,TESTING AND FINDING ACCURACY (MODEL ANALYSIS)

A. MODEL TRAINING:

In this section, we will be creating and training our model for predicting what a car's mpg will be. Since there are multiple algorithms we can use to build our model, we will compare the accuracy scores after testing and pick the most accurate algorithm.

Now, let's initiate our different regression algorithms. Then, we train them and check the accuracy on the training set.

We also want to partition our dataset into training, testing, and validation, so let's add a method for that ability.

Training is first, and it's where our model generates "intuition" about how to approach fraudulent and not fraudulent transactions.

Let's perform the splitting of our data into test, train, validation using `train_test_split`.

It is similar to a student studying and developing knowledge about a topic before an exam.

The testing phase is where we see how the model performs against data where we know the outcome. This would be the exam if we continue the analogy from before.

The algorithms will perform differently, similar to how students will score differently on the exam. From this phase, we generate an accuracy score to compare the different algorithms.

B. TESTING THE MODEL:

We are doing regression testing, we use sklearn's built-in `mean_absolute_error()` method and then use `mean_absolute_percentage_error()` to get percentage error.

Model-based testing is a software testing technique where the run time behavior of the software under test is checked against predictions made by a model. A model is a description of a system's behavior

Defining `DecisionTreeRegressor()` as `tree_model` and fitting trained values, to predict model score for testing done by `DecisionTree`.

Defining `RandomForestRegressor()` as `tree_model` and fitting trained values, to predict model score for testing done by `RandomForest`.

Training and Testing the Data

```
In [20]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 1)
```

```
In [21]: print(X_train.shape)
print(X_test.shape)

(298, 7)
(100, 7)
```

```
In [22]: print(y_train.shape)
print(y_test.shape)

(298,)
(100,)
```

```
In [23]: tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)

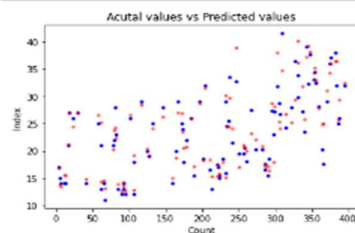
tree_r2 = tree_model.score(X_test, y_test)
```

```
In [24]: model = RandomForestRegressor()
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

```
In [25]: neap = mean_absolute_percentage_error(y_test, pred)
print(neap)

0.07944477552947934
```

```
In [26]: plt.scatter(X_test.index, y_test, marker=".", color="blue")
plt.scatter(X_test.index, pred, marker=".", color="red", alpha=0.5)
plt.title("Acutal values vs Predicted values")
plt.xlabel("Count")
plt.ylabel("Index")
plt.show()
```



TAB 4.1– Training and Testing the Data

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 REFERENCE

- [1] **Srivatsa Bindingnolle Narshima**, "Prediction Modeling Of Fuel Efficiency Of Trucks", IPTEK Journal Preceedings Series, 1(1): Pp.224-229, 2015.
- [2] **Hyunkun kim Hyeongoo Pyeon**, Jong Sool Park, Jin Young Hwang, Sejoon Lim 3, "Autonomous Vehicle Fuel Economy Optimization with Deep Reinforcement Learning", Doi:10.3390/Electronics9111911, 2020, 9, 1911.
- [3] **Sandareka K.W.R.S. Wickramanayake** And **H.M.N Dilum Bandara**, "Fuel Consumption Prediction Of Fleet Vehicles Using Machine Learning: A comparative Study", 978-1- 4799-1740- 2/15/\$31.00, 2015 IEEE.
- [4] **Ramadoni Syahputra**, "Application of Neuro-Fuzzy Method for Prediction of Vehicle Fuel Consumption", 10th April 2016. Vol.86. No.1, 2005 - 2016 JATIT & LLS
- [5] **Mohd Saifullah, Rusiman Efendi, Nasibov Robiah Adnan**, "The Optimal Fuzzy C-Regression Models (OFCRM) Tn Miles Per Gallon of Cars Prediction", 2011 IEEE Student Conference On Research And Development.

5.2 APENDIX

CONCLUSION:

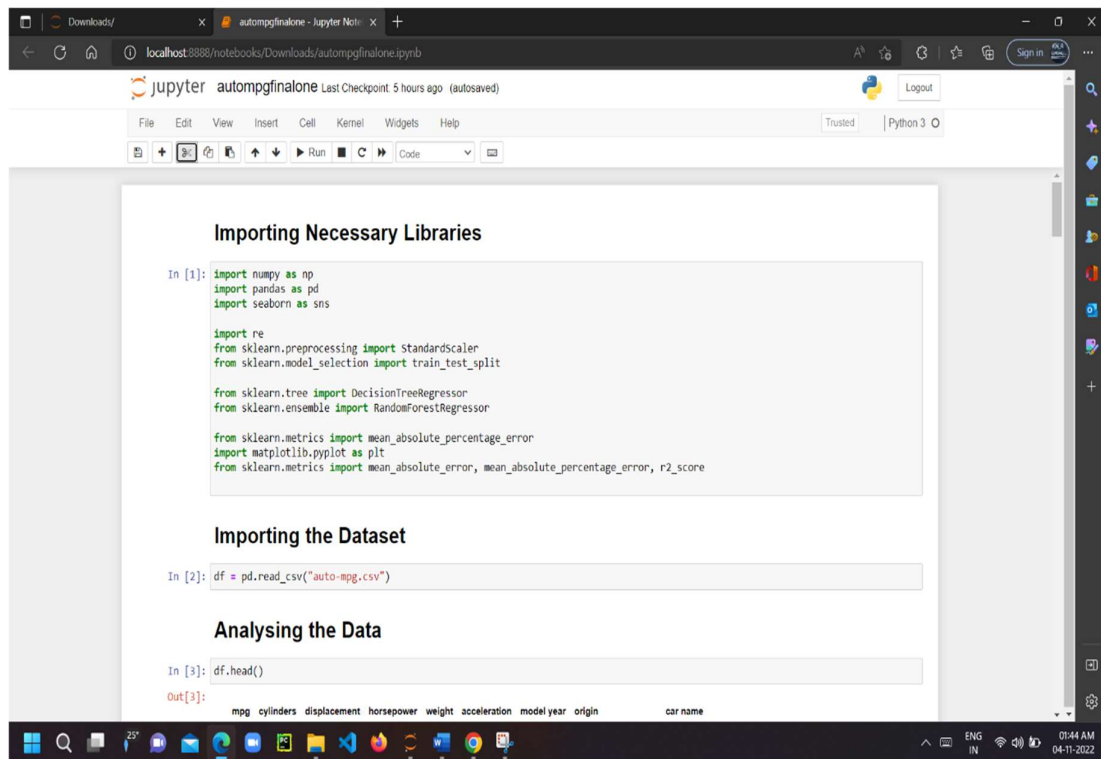
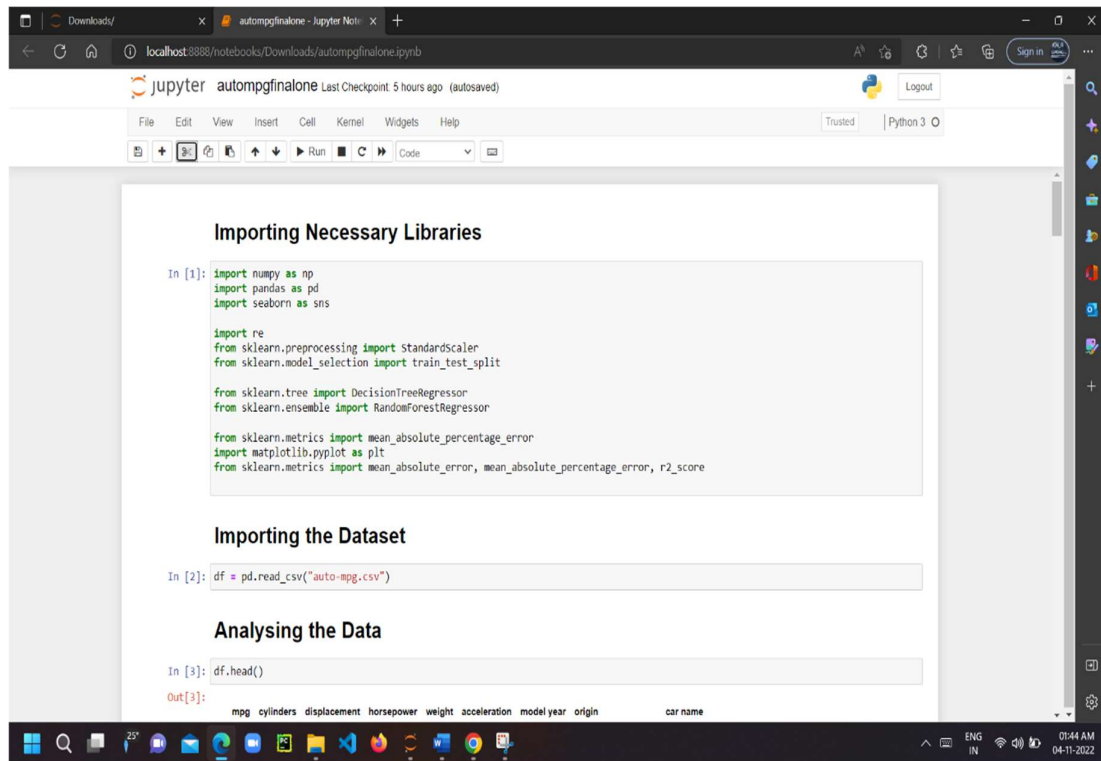
This model could be trained with newer car data and be used to predict, allowing companies to potentially resources currently used on R&D today on making more efficient, more popular vehicles that outshine competitors.

While our model may be inaccurate in some cases, we talked about how our dataset can contain inaccurate values for the mpg, and oftentimes, our predictions are more accurate than the values in the dataset.

For newer cars, the collected data is significantly more reliable, so our model will be able to perform better with a different, more accurate dataset.

If you want to optimize spending your resources and outshine the competition.

A. SCREENSHOTS:



Downloads/ x autompgfinalone - Jupyter Note x +

localhost:8888/notebooks/Downloads/autompgfinalone.ipynb

jupyter autompgfinalone Last Checkpoint: 5 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

Analysing the Data

```
In [3]: df.head()
```

```
Out[3]:
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3690	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

```
In [4]: # comments
# col_names = ["mpg","cylinders","displacement","horsepower","weight","acceleration","model year","origin","car name"]
# df = pd.read_csv("autompg.csv",names = col_names)
```

```
In [5]: df.columns
```

```
Out[5]: Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'model year', 'origin', 'car name'],
dtype='object')
```

Checking the Null Values

```
In [6]: df.isnull().sum()
```

```
Out[6]: mpg      0
cylinders  0
displacement 0
horsepower  0
```

Downloads/ x autompgfinalone - Jupyter Note x +

localhost:8888/notebooks/Downloads/autompgfinalone.ipynb

jupyter autompgfinalone Last Checkpoint: 5 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

Checking the Null Values

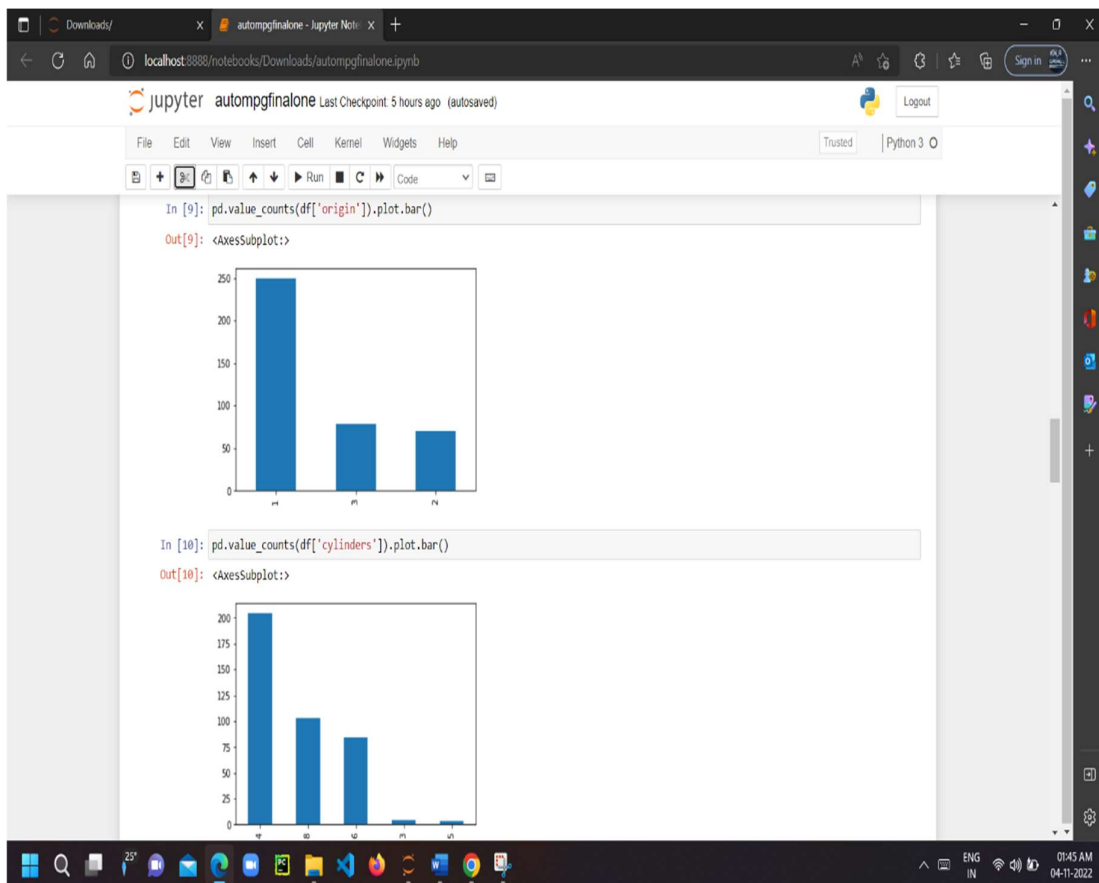
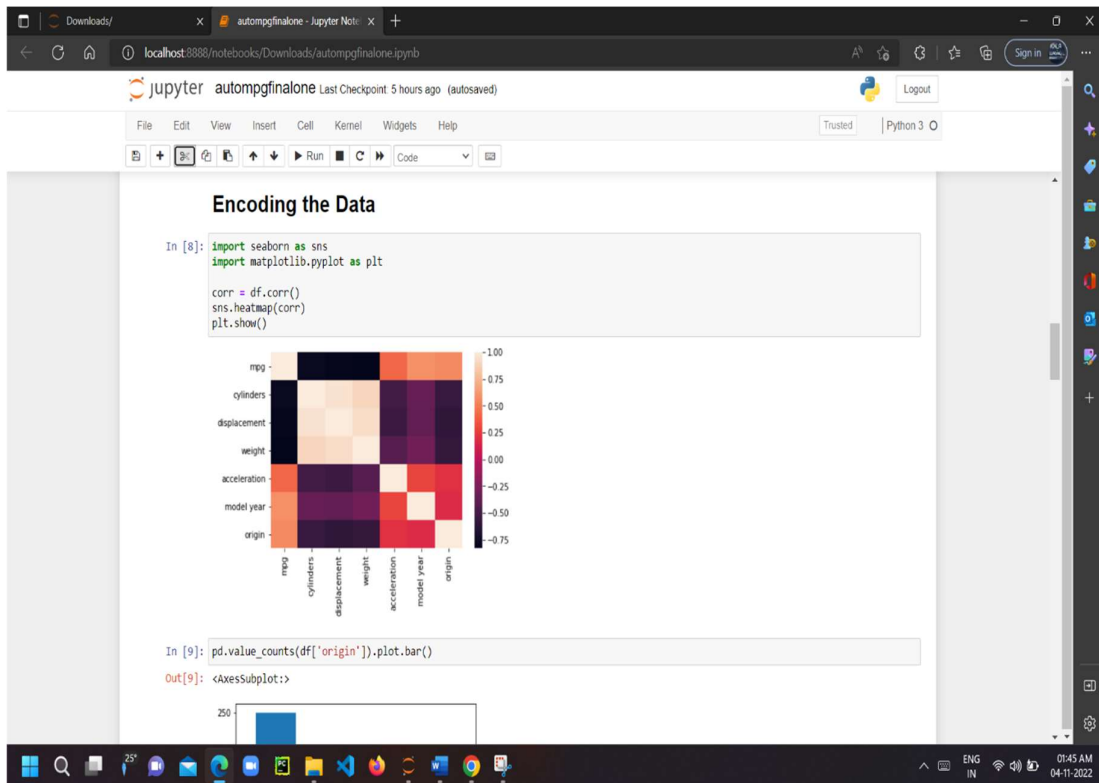
```
In [6]: df.isnull().sum()
```

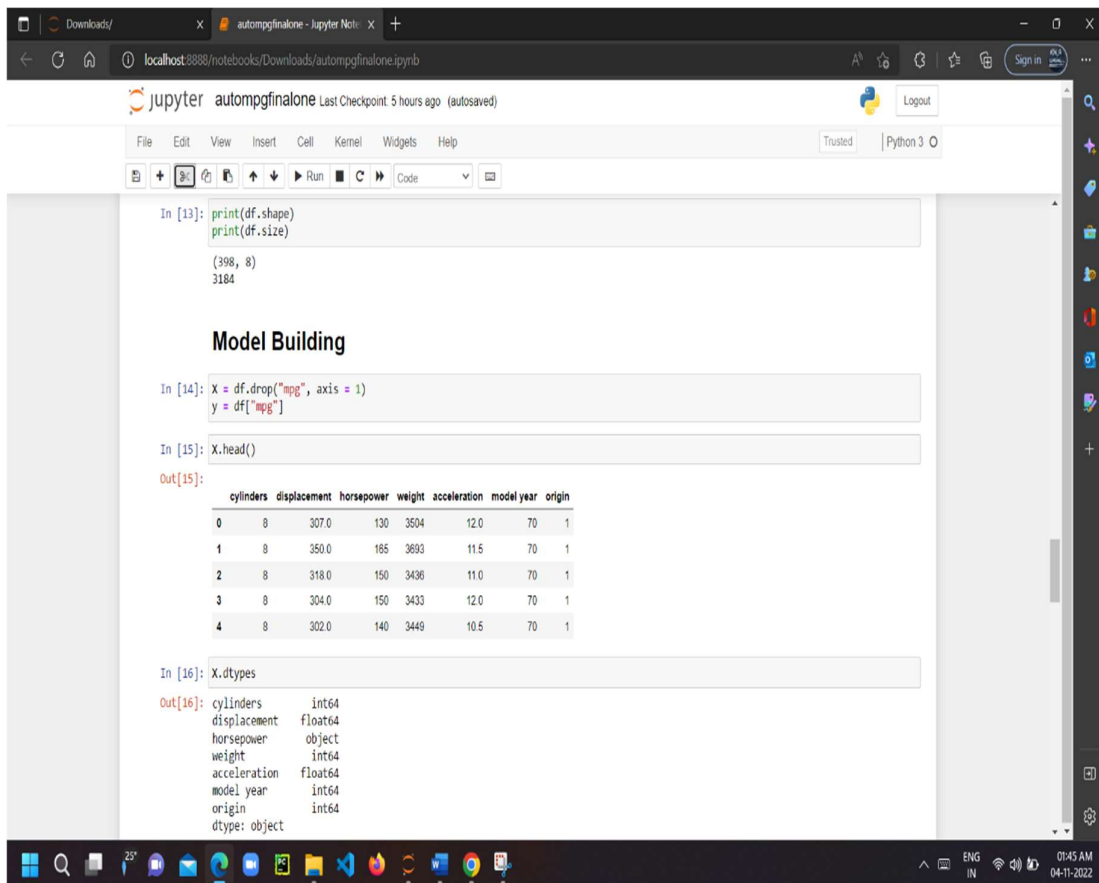
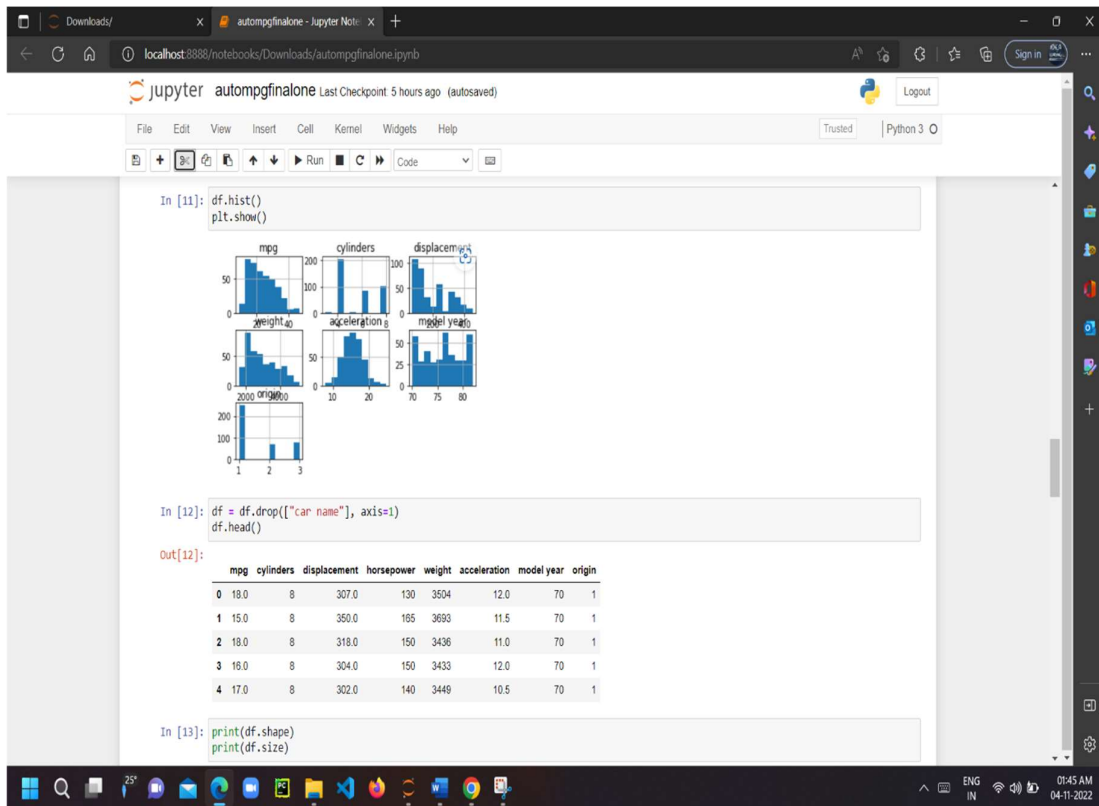
```
Out[6]: mpg      0
cylinders  0
displacement 0
horsepower  0
weight      0
acceleration 0
model year  0
origin      0
car name    0
dtype: int64
```

```
In [7]: df.describe()
```

```
Out[7]:
```

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425079	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697927	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000





```
model year    int64
origin        int64
dtype: object

In [17]: # for row in X["horsepower"]:
#         if type(row) == int:
#             pass
#         else:
#             print(row, type(row))

In [18]: # for row in X["horsepower"]:
#         if row == "?":
#             row = 0
#         else:
#             row = int(row)
for i in range(len(X["horsepower"])):
    if X["horsepower"][i] == "?":
        X["horsepower"][i] = 0
    else:
        X["horsepower"][i] = int(X["horsepower"][i])

<ipython-input-18-198a0b2eef2d>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
X["horsepower"][i] = int(X["horsepower"][i])
<ipython-input-18-198a0b2eef2d>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
X["horsepower"][i] = 0
```

```
In [19]: y.head()
Out[19]: 0    18.0
1    15.0
2    18.0
3    16.0
4    17.0
Name: mpg, dtype: float64

Training and Testing the Data

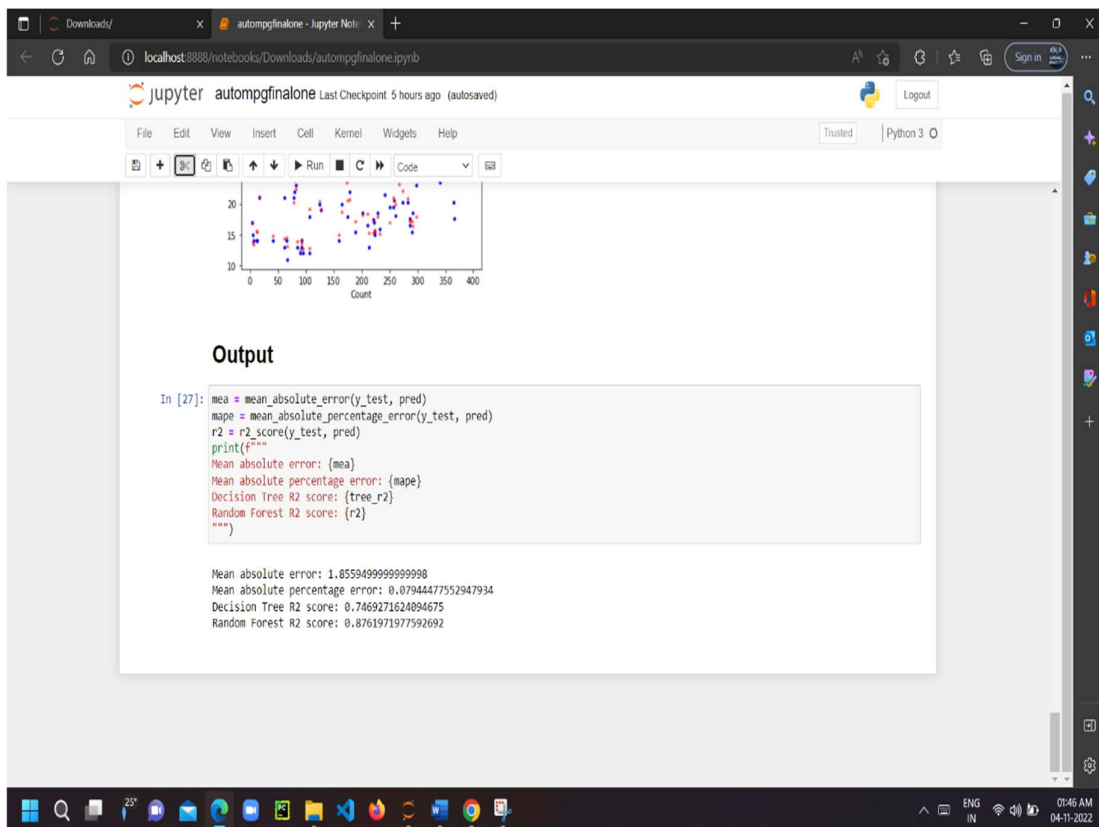
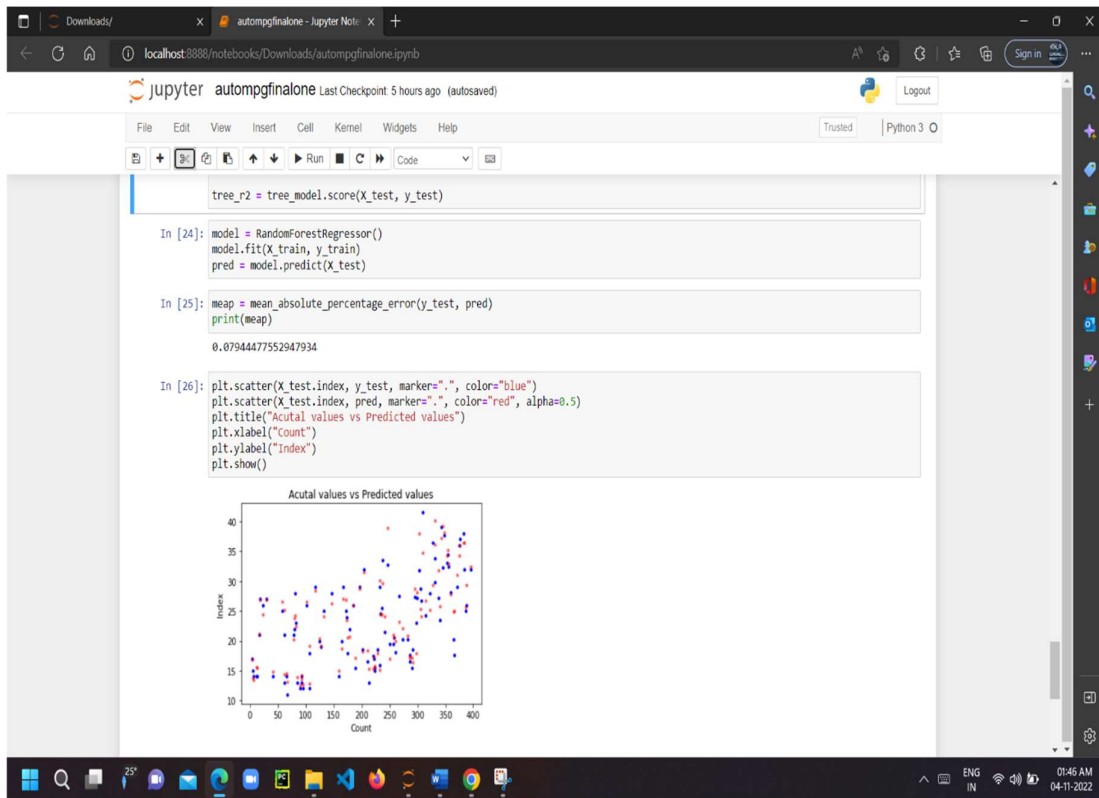
In [20]: x_train, x_test, y_train, y_test = train_test_split(X, y, random_state = 1)

In [21]: print(X_train.shape)
print(X_test.shape)
(298, 7)
(100, 7)

In [22]: print(y_train.shape)
print(y_test.shape)
(298,)
(100,)

In [23]: tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
tree_r2 = tree_model.score(X_test, y_test)

In [24]: model = RandomForestRegressor()
model.fit(X_train, y_train)
pred = model.predict(X_test)
```



B. SOURCE CODE:

```
# Importing Necessary Libraries
```

```
import numpy as np
```

```
import pandas as pd
```

```
import seaborn as sns
```

```
import re
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.metrics import mean_absolute_percentage_error
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import mean_absolute_error,  
mean_absolute_percentage_error, r2_score
```

```
# Importing the Dataset
```

```
df = pd.read_csv("auto-mpg.csv")
```

```
# Analysing the Data
```

```
df.head()
```

```
# comments
```

```
#                                col_names                                =  
["mpg","cylinders","displacement","horsepower","weight","acceleration","model  
year","origin","car name"]
```

```
# df = pd.read_csv("autompg.csv",names = col_names)
```

```
df.columns
```

```
# Checking the Null Values
```

```
df.isnull().sum()
```

```
df.describe()
```

```
# Encoding the Data
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
corr = df.corr()
```

```
sns.heatmap(corr)
```

```
plt.show()
```

```
pd.value_counts(df['origin']).plot.bar()
```

```
pd.value_counts(df['cylinders']).plot.bar()
```

```
df.hist()
```

```
plt.show()
```

```
df = df.drop(["car name"], axis=1)
```

```
df.head()
```

```
print(df.shape)
```

```
print(df.size)
```

```
# Model Building
```

```
X = df.drop("mpg", axis = 1)
```

```
y = df["mpg"]
```

```
X.head()
```

```
X.dtypes
```

```
# for row in X["horsepower"]:
```

```

# if type(row) == int:

#     pass

# else:

#     print(row, type(row))


# for row in X["horsepower"]:

#     if row == "?":

#         row = 0

#     else:

#         row = int(row)

for i in range(len(X["horsepower"])):

    if X["horsepower"][i] == "?":

        X["horsepower"][i] = 0

    else:

        X["horsepower"][i] = int(X["horsepower"][i])


y.head()


# Training and Testing the Data


X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 1)

```

```
print(X_train.shape)
```

```
print(X_test.shape)
```

```
print(y_train.shape)
```

```
print(y_test.shape)
```

```
tree_model = DecisionTreeRegressor()
```

```
tree_model.fit(X_train, y_train)
```

```
tree_r2 = tree_model.score(X_test, y_test)
```

```
model = RandomForestRegressor()
```

```
model.fit(X_train, y_train)
```

```
pred = model.predict(X_test)
```

```
meap = mean_absolute_percentage_error(y_test, pred)
```

```
print(meap)
```

```
plt.scatter(X_test.index, y_test, marker=".", color="blue")
```

```
plt.scatter(X_test.index, pred, marker=".", color="red", alpha=0.5)
```

```
plt.title("Actual values vs Predicted values")
```

```
plt.xlabel("Count")
```

```
plt.ylabel("Index")
```

```
plt.show()
```

```
# Output
```

```
mea = mean_absolute_error(y_test, pred)

mape = mean_absolute_percentage_error(y_test, pred)

r2 = r2_score(y_test, pred)

print(f"""

Mean absolute error: {mea}

Mean absolute percentage error: {mape}

Decision Tree R2 score: {tree_r2}

Random Forest R2 score: {r2}

""")
```