|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Date** | **Practical Description** | **Sign** |
| 1a |  | **Explore existing Packages, API's, Data Sets and Models**  Explore Jupyter Notebook, RStudio, Explore Google Colab for Python/R |  |
| 1b |  | **Data Science Methodology -** Problem to Approach, Requirements to Collection, Understanding to Preparation, Modelling to Evaluation, Deployment to Feedback |  |
| 2a |  | **Numpy -** Arrays, Dimensions- 2D, 3D, ND, Broadcasting, Indexing, Slicing, Numpy Functions: array manipulation, string, arithmetic, statistical, Numpy Functions: arrange, linspace, random number generation, seed, reshape, ravel |  |
| 2b |  | **Pandas -** Series functions: empty, ndim, size, dtype, values, head, tail, DataFrame functions: datatype, transpose, empty, ndim, shape, size, values, head, tail, DateTime |  |
| 2c |  | **Matplotlib, Seaborn-**Plyplot, plotting, markers, line, labels, grid, subplot, scatter, bar, histogram, pie charts, countplot |  |
| 3a |  | **Data Analysis -** Import Libraries & datasets, load data, check data type, shape, drop columns, merge datasets, sort, concatenate, statistical summary of data, skewness, co-relation |  |
| 3b |  | **Data Wrangling** - Pre-processing Data - Dealing with Missing values, Correcting Data Format, Data standardization, Data Normalization, Binning, Turning categorical variables into quantitative variables |  |
| 4 |  | **Exploratory Data Analysis** Descriptive Statistics, Categorical Variables - box plots, Value Counts, Scatterplot , Group By, Pivot Table, Heat Map, |  |
| 5 |  | Model Evaluation using Visualization, Measures for In-Sample Evaluation |  |
| 6 |  | Data Visualization using Tableau |  |
| 7 |  | Regression – Linear, Logistic |  |
| 8 |  | Classification, Decision Trees, Random Forest |  |
| 9 |  | Clustering, Types, Optimal number of clusters |  |

**Practical 1A**

**Aim - Explore existing Packages, API's, Data Sets and Models, Explore Jupyter Notebook, RStudio, Explore Google Colab for Python/R**

**Softwares used:**

**Theory:**

**1. Exploring Existing Python packages for machine learning**:

1. [Matplotlib: An interactive plotting library for creating high-quality graphs and charts](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
2. [Natural Language Toolkit (NLTK): A framework for developing both symbolic and statistical Natural Language Processing (NLP) in Python](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
3. **Numpy**: A fundamental package for numerical computations and handling arrays.
4. [Scipy: A library for scientific and technical computing, including optimization, integration, and linear algebra](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
5. [Scikit-learn: A powerful machine learning library for classification, regression, clustering, and more](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
6. **Theano**: A numerical computation library for deep learning models (though it’s no longer actively developed).
7. [TensorFlow: An open-source machine learning framework developed by Google, widely used for deep learning applications](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
8. [Keras: A high-level neural networks API that runs on top of TensorFlow or Theano, simplifying deep learning model development](https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/" \t "https://www.bing.com/_blank).
9. [PyTorch: A dynamic deep learning framework with a strong focus on research and flexibility](https://www.activestate.com/blog/top-10-python-machine-learning-packages/).
10. [Pandas: A data manipulation library for handling structured data, including dataframes and series](https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/" \t "https://www.bing.com/_blank).

**Exploring Existing R packages for machine learning**:

1. data.table: A high-performance package that enhances R's data.frame, providing efficient data manipulation, parallelism, and feature-rich joins.
2. dplyr: A powerful data manipulation package with five key verbs (Select, Filter, Arrange, Mutate, and Summarize) for efficient data wrangling.
3. ggplot2: A popular package for creating elegant and customizable data visualizations.
4. caret: A comprehensive package for building and evaluating machine learning models, including preprocessing, feature selection, and cross-validation.
5. e1071: Provides functions for support vector machines, naive Bayes, and other statistical learning methods.
6. xgboost: A gradient boosting library for efficient tree-based ensemble models.
7. randomForest: Implements random forests, an ensemble learning method based on decision trees.

These packages play crucial roles in various aspects of machine learning, from data preprocessing to model building and evaluation. Each package has its own strengths and use cases, choosing the right one depends on the specific project requirements.

**Exploring Application Programming Interface(API)**

An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other. It defines how requests and responses should be structured, enabling seamless interaction between systems.

* APIs serve as bridges, enabling developers to access specific functionalities or data from external services, libraries, or platforms.

**Machine Learning APIs:**

In the context of machine learning, an ML API provides a way to interact with trained models, making predictions, classifications, or other data-driven tasks. Here are some key points about ML APIs:

* Remote Access: An ML API allows you to access machine learning capabilities remotely. Instead of running models locally, you can send data to the API, which processes it and returns results.
* Problem-Specific Solutions: ML APIs are tailored to solve specific problems within a project. For instance, you might use an API to perform sentiment analysis, image recognition, or recommendation tasks.
* Input-Output Interaction: Users provide input data (such as text, images, or numerical features) to the API, which then processes it using the underlying machine learning model. The API responds with predictions, classifications, or other relevant information.
* Pattern Matching: ML APIs often involve pattern matching. Given input data, the system identifies patterns based on the trained model and provides relevant insights.

**Use Cases for ML APIs:**

ML APIs find applications in various domains. Here are a few examples:

* Natural Language Processing (NLP): APIs like Google's Cloud Natural Language API analyze text sentiment, extract entities, and perform language detection.
* Image Recognition: APIs such as Google Vision and Amazon Rekognition identify objects, faces, and scenes in images.
* Speech-to-Text: APIs like Google Cloud Speech-to-Text convert spoken language into written text.
* Translation: APIs handle language translation, making it easier to build multilingual applications.
* Recommendation Systems: APIs can power personalized recommendations for users based on their behavior.

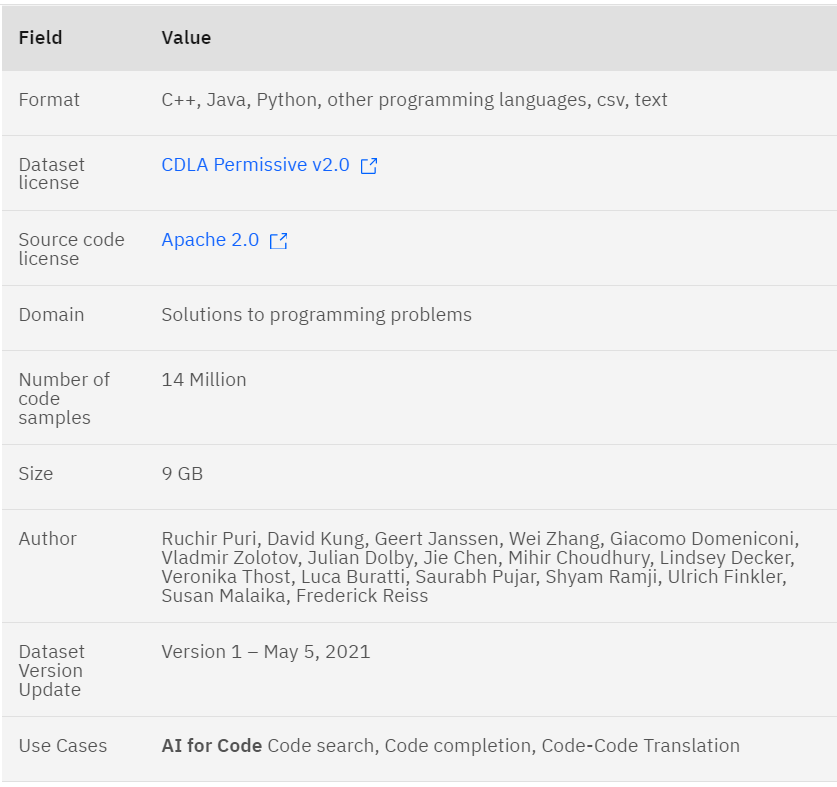
**FastAPI for Deploying ML Models:**

* FastAPI is a Python web framework that excels at creating APIs. Its features:
* Speed: FastAPI is faster than traditional frameworks like Flask.
* Data Validation: Unlike Flask, FastAPI provides easy data validation, allowing you to define specific data types for input.
* Automatic Docs: FastAPI generates interactive API documentation (Swagger UI and Redoc) automatically.
* Async Support: It supports asynchronous programming, making it efficient for handling multiple requests.
* GraphQL and Websockets: FastAPI includes built-in support for these technologies.

**Exploring Datasets**

**Source: IBM <https://developer.ibm.com/exchanges/data>**

**CodeNet : CodeNet is a large-scale dataset with approximately 14 million code samples, each of which is an intended solution to one of 4000 coding problems.**



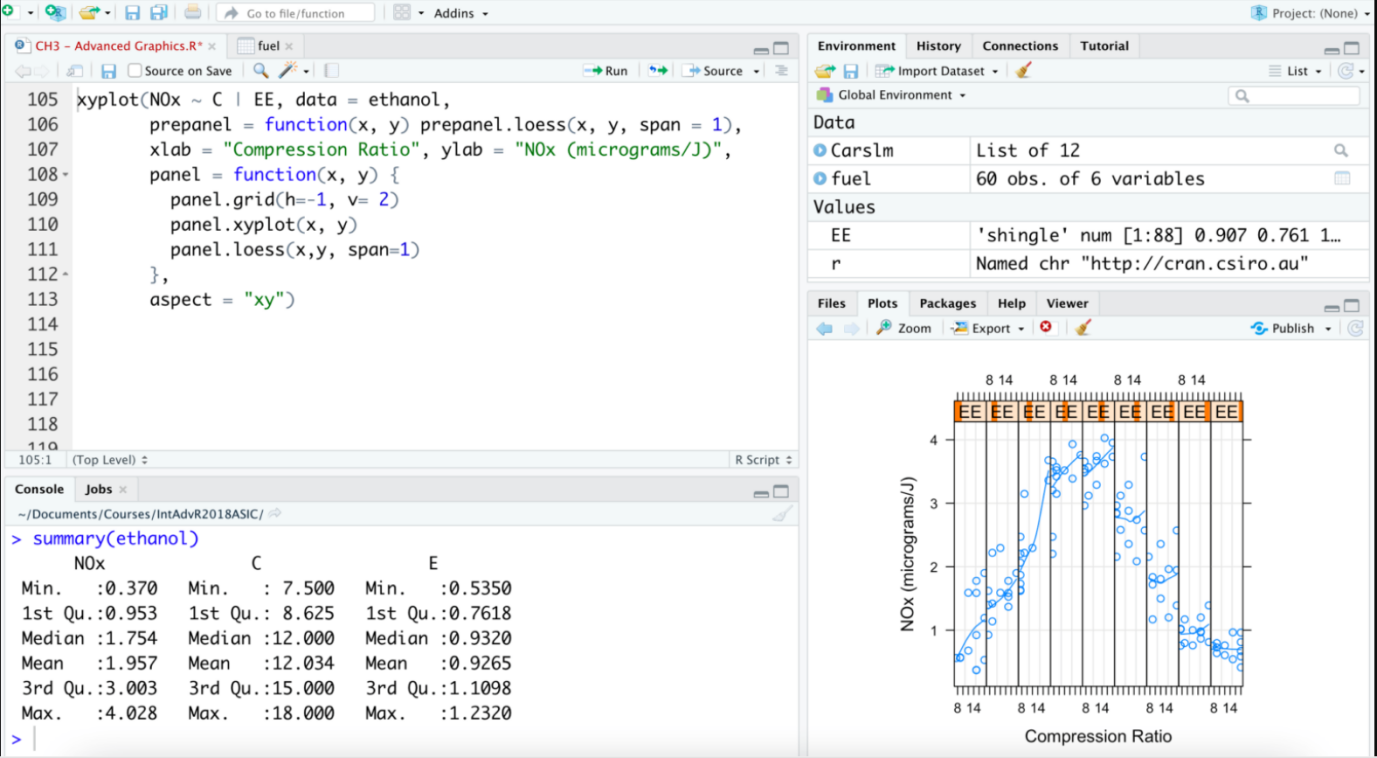
Source: <https://data.gov.in/high-value-datasets>



**Explore Jupyter Notebook**



**Explore RStudio**



**Explore Google Colab for Python/R**

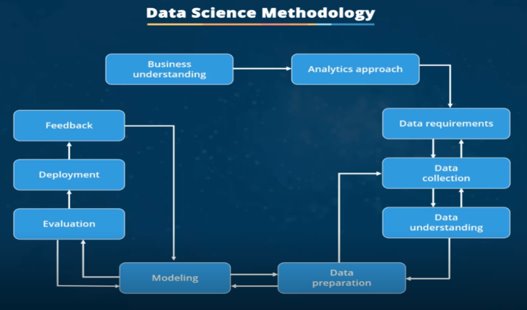


**Conclusion :** The practical to explore existing packages, API's, Data Sets and Models, Jupyter Notebook, RStudio, Google Colab for Python/R has been successfully completed.

**Practical 1B**

**Aim - Study Data Science Methodology - Problem to Approach, Requirements to Collection, Understanding to Preparation, Modelling to Evaluation, Deployment to Feedback**

Case Study: What is the best way to allocate the limited healthcare budget to maximize its use in providing quality care?

****

This question is one that became a hot topic for an American healthcare insurance provider. As public funding for readmissions was decreasing, this insurance company was at risk of having to make up for the cost difference, which could potentially increase rates for its customers.

Knowing that raising insurance rates was not going to be a popular move, the insurance company sat down with the health care authorities in its region and brought in IBM data scientists to see how data science could be applied to the question at hand.

**Business Understanding: Problem to Approach**

* Before even starting to collect data, the goals and objectives needed to be defined.
* After spending time to determine the goals and objectives, the team prioritized "patient readmissions" as an effective area for review.
* With the goals and objectives in mind, it was found that approximately 30% of individuals who finish rehab treatment would be readmitted to a rehab center within one year; and that 50% would be readmitted within five years.
* After reviewing some records, it was discovered that the patients with congestive heart failure were at the top of the readmission list.
* It was further determined that a decision-tree model could be applied to review this scenario, to determine why this was occurring.
* To gain the business understanding that would guide the analytics team in formulating and performing their first project, the IBM Data scientists, proposed and delivered an on-site workshop to kick things off.
* The key business sponsors involvement throughout the project was critical, in that the sponsor:
  + Set overall direction
  + Remained engaged and provided guidance.
  + Ensured necessary support, where needed.
* Finally, four business requirements were identified for whatever model would be built.
  1. Predicting readmission outcomes for those patients with Congestive Heart Failure
  2. Predicting readmission risk.
  3. Understanding the combination of events that led to the predicted outcome
  4. Applying an easy-to-understand process to new patients, regarding their readmission risk.

**Analytic Approach**

* For the case study, a decision tree classification model was used to identify the combination of conditions leading to each patient's outcome.
* In this approach, examining the variables in each of the nodes along each path to a leaf, led to a respective threshold value.
* This means the decision tree classifier provides both the predicted outcome, as well as the likelihood of that outcome, based on the proportion at the dominant outcome, yes or no, in each group.
* From this information, the analysts can obtain the readmission risk, or the likelihood of a yes for each patient. If the dominant outcome is yes, then the risk is simply the proportion of yes patients in the leaf.
* If it is no, then the risk is 1 minus the proportion of no patients in the leaf.
* A decision tree classification model is easy for non-data scientists to understand and apply, to score new patients for their risk of readmission.
* Clinicians can readily see what conditions are causing a patient to be scored as high-risk and multiple models can be built and applied at various points during hospital stay.
* This gives a moving picture of the patient's risk and how it is evolving with the various treatments being applied. For these reasons, the decision tree classification approach was chosen for building the Congestive Heart Failure readmission model.

**From Requirements to Collection**

* Collecting data requires that you know the source or, know where to find the data elements that are needed.
* In the context of our case study, these can include: demographic, clinical and coverage information of patients, provider information, claims records, as well as pharmaceutical and other information related to all the diagnoses of the congestive heart failure patients.
* For this case study, certain drug information was also needed, but that data source was not yet integrated with the rest of the data sources.
* This leads to an important point: It is alright to defer decisions about unavailable data, and attempt to acquire it at a later stage. For example, this can even be done after getting some intermediate results from the predictive modeling.
* If those results suggest that the drug information might be important in obtaining a good model, then the time to try to get it would be invested.
* As it turned out though, they were able to build a reasonably good model without this drug information.
* DBAs and programmers often work together to extract data from various sources, and then merge it.
* This allows for removing redundant data, making it available for the next stage of the methodology, which is data understanding.
* At this stage, if necessary, data scientists and analytics team members can discuss various ways to better manage their data, including automating certain processes in the database, so that data collection is easier and faster.

**Data Understanding: From Understanding to Preparation**

* Data understanding encompasses all activities related to constructing the data set.
* Essentially, the data understanding section of the data science methodology answers the question: Is the data that you collected representative of the problem to be solved?
* In order to understand the data related to congestive heart failure admissions, descriptive statistics needed to be run against the data columns that would become variables in the model.
* First, these statistics included Hearst, univariates, and statistics on each variable, such as mean, median, minimum, maximum, and standard deviation.
* Second, pairwise correlations were used, to see how closely certain variables were related, and which ones, if any, were very highly correlated, meaning that they would be essentially redundant, thus making only one relevant for modeling.
* Third, histograms of the variables were examined to understand their distributions. Histograms are a good way to understand how values or a variable are distributed, and which sorts of data preparation may be needed to make the variable more useful in a model. For example, for a categorical variable that has too many distinct values to be informative in a model, the histogram would help them decide how to consolidate those values.
* The univariates, statistics, and histograms are also used to assess data quality.
* From the information provided, certain values can be re-coded or perhaps even dropped if necessary, such as when a certain variable has many missing values.
* The question then becomes, does "missing" mean anything?
* Sometimes a missing value might mean "no", or "0" (zero), or at other times it simply means "we don't know". Or, if a variable contains invalid or misleading values, such as a numeric variable called "age" that contains 0 to 100 and also 999, where that "triple-9" actually means "missing", but would be treated as a valid value unless we corrected it.
* Initially, the meaning of congestive heart failure admission was decided on the basis of a primary diagnosis of congestive heart failure.
* But working through the data understanding stage revealed that the initial definition was not capturing all of the congestive heart failure admissions that were expected, based on clinical experience.
* This meant looping back to the data collection stage and adding secondary and tertiary diagnoses, and building a more comprehensive definition of congestive heart failure admission.
* This is just one example of the interactive processes in the methodology.
* The more one works with the problem and the data, the more one learns and therefore the more refinement that can be done within the model, ultimately leading to a better solution to the problem.

**Data Preparation: Understanding to Preparation**

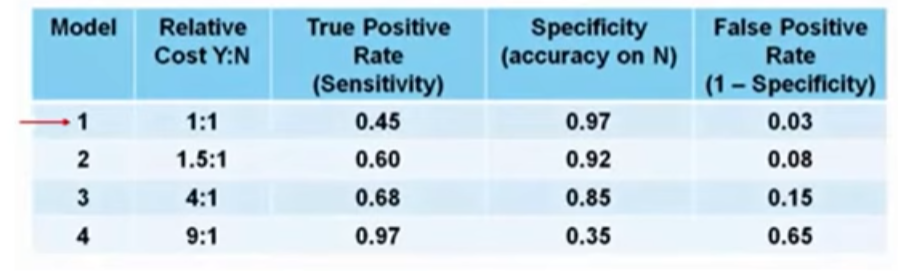
* Together with data collection and data understanding, data preparation is the most time-consuming phase of a data science project, typically taking seventy percent and even up to even ninety percent of the overall project time.
* Automating some of the data collection and preparation processes in the database, can reduce this time to as little as 50 percent. This time savings translates into increased time for data scientists to focus on creating models.
* Transforming data in the data preparation phase is the process of getting the data into a state where it may be easier to work with. Specifically, the data preparation stage of the methodology answers the question: What are the ways in which data is prepared?
* To work effectively with the data, it must be prepared in a way that addresses missing or invalid values and removes duplicates, toward ensuring that everything is properly formatted.
* Feature engineering is also part of data preparation. It is the process of using domain knowledge of the data to create features that make the machine learning algorithms work.
* A feature is a characteristic that might help when solving a problem. Features within the data are important to predictive models and will influence the results you want to achieve.
* Feature engineering is critical when machine learning tools are being applied to analyze the data.
* When working with text, text analysis steps for coding the data are required to be able to manipulate the data.
* The data scientist needs to know what they're looking for within their dataset to address the question.
* The text analysis is critical to ensure that the proper groupings are set, and that the programming is not overlooking what is hidden within.
* The data preparation phase sets the stage for the next steps in addressing the question.
* While this phase may take a while to do, if done right the results will support the project. If this is skipped over, then the outcome will not be up to par and may have you back at the drawing board.
* It is vital to take your time in this area, and use the tools available to automate common steps to accelerate data preparation.
* We need to make sure to pay attention to the detail in this area. After all, it takes just one bad ingredient to ruin a fine meal.

**Modeling: From Modeling to Evaluation**

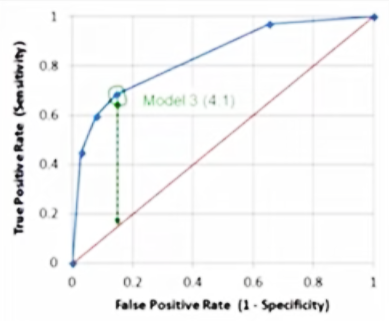
* Modelling is the stage in the data science methodology where the data scientist has the chance to sample the model created and determine if it's bang on or in need of more rework!
* This portion of the course is geared toward answering two key questions:
* First, what is the purpose of data modeling, and second, what are some characteristics of this process?
* Data Modelling focuses on developing models that are either descriptive or predictive.
* An example of a descriptive model might examine things like: if a person did this, then they're likely to prefer that.
* A predictive model tries to yield yes/no, or stop/go type outcomes.
* These models are based on the analytic approach that was taken, either statistically driven or machine learning driven.
* The data scientist will use a training set for predictive modelling.
* A training set is a set of historical data in which the outcomes are already known. The training set acts like a gauge to determine if the model needs to be calibrated.
* In this stage, the data scientist will play around with different algorithms to ensure that the variables in play are actually required.
* The success of data compilation, preparation and modelling, depends on the understanding of the problem at hand, and the appropriate analytical approach being taken.
* The data supports the answering of the question, and like the quality of the ingredients in cooking, sets the stage for the outcome.
* Constant refinement, adjustments and tweaking are necessary within each step to ensure the outcome is one that is solid.
* In John Rollins' descriptive Data Science Methodology, the framework is geared to do 3 things:
  + First, understand the question at hand.
  + Second, select an analytic approach or method to solve the problem, and
  + third, obtain, understand, prepare, and model the data.
* The end goal is to move the data scientist to a point where a data model can be built to answer the question.
* In this stage of the methodology, model evaluation, deployment, and feedback loops ensure that the answer is near and relevant.
* This relevance is critical to the data science field overall, as it is a fairly new field of study, and we are interested in the possibilities it has to offer.

**Evaluation: From Modeling to Evaluation**

* A model evaluation goes hand-in-hand with model building as such, the modeling and evaluation stages are done iteratively. Model evaluation is performed during model development and before the model is deployed.
* Evaluation allows the quality of the model to be assessed but it's also an opportunity to see if it meets the initial request. Evaluation answers the question: Does the model used really answer the initial question or does it need to be adjusted?
* Model evaluation can have two main phases.
  + The first is the diagnostic measures phase, which is used to ensure the model is working as intended. If the model is a predictive model, a decision tree can be used to evaluate if the answer the model can output, is aligned to the initial design. It can be used to see where there are areas that require adjustments. If the model is a descriptive model, one in which relationships are being assessed, then a testing set with known outcomes can be applied, and the model can be refined as needed.
  + The second phase of evaluation that may be used is statistical significance testing. This type of evaluation can be applied to the model to ensure that the data is being properly handled and interpreted within the model. This is designed to avoid unnecessary second guessing when the answer is revealed.
* So now, let's go back to our case study so that we can apply the "Evaluation" component within the data science methodology.
* Let's look at one way to find the optimal model through a diagnostic measure based on tuning one of the parameters in model building.
* Specifically, we'll see how to tune the relative cost of misclassifying yes and no outcomes.



* As shown in this table, four models were built with four different relative misclassification costs.
* As we see, each value of this model-building parameter increases the true-positive rate, or sensitivity, of the accuracy in predicting yes, at the expense of lower accuracy in predicting no, that is, an increasing false-positive rate.
* The question then becomes, which model is best based on tuning this parameter?
* For budgetary reasons, the risk-reducing intervention could not be applied to most or all congestive heart failure patients, many of whom would not have been readmitted anyway.
* On the other hand, the intervention would not be as effective in improving patient care as it should be, with not enough high-risk congestive heart failure patients targeted.
* So, how do we determine which model was optimal?



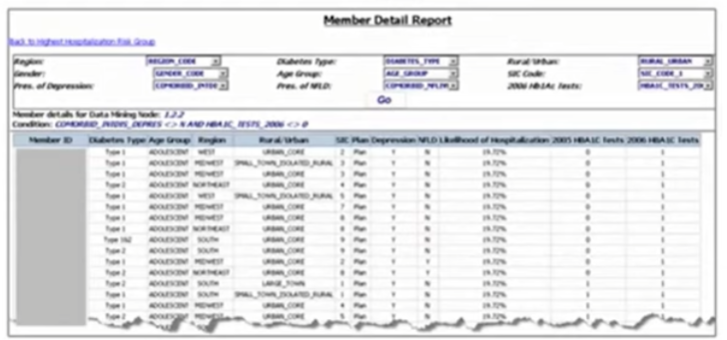
* The optimal model is the one giving the maximum separation between the blue ROC curve relative to the red base line.
* We can see that model 3, with a relative misclassification cost of 4-to-1, is the best of the 4 models.
* And just in case you were wondering, ROC stands for receiver operating characteristic curve, which was first developed during World War II to detect enemy aircraft on radar. It has since been used in many other fields as well.
* Today it is commonly used in machine learning and data mining.
* The ROC curve is a useful diagnostic tool in determining the optimal classification model. This curve quantifies how well a binary classification model performs, declassifying the yes and no outcomes when some discrimination criterion is varied. In this case, the criterion is a relative misclassification cost.
* By plotting the true-positive rate against the false-positive rate for different values of the relative misclassification cost, the ROC curve helped in selecting the optimal model.

**Deployment: From Deployment to Feedback**

* In preparation for solution deployment, the next step was to assimilate the knowledge for the business group who would be designing and managing the intervention program to reduce readmission risk.
* In this scenario, the business people translated the model results so that the clinical staff could understand how to identify high-risk patients and design suitable intervention actions.
* The goal, of course, was to reduce the likelihood that these patients would be readmitted within 30 days after discharge.
* During the business requirements stage, the Intervention Program Director and her team had wanted an application that would provide automated, near real-time risk assessments of congestive heart failure.
* It also had to be easy for clinical staff to use, and preferably through browser-based application on a tablet, that each staff member could carry around.
* This patient data was generated throughout the hospital stay. It would be automatically prepared in a format needed by the model and each patient would be scored near the time of discharge.
* Clinicians would then have the most up-to-date risk assessment for each patient, helping them to select which patients to target for intervention after discharge.
* As part of solution deployment, the Intervention team would develop and deliver training for the clinical staff.
* Also, processes for tracking and monitoring patients receiving the intervention would have to be developed in collaboration with IT developers and database administrators, so that the results could go through the feedback stage and the model could be refined over time.



* This map is an example of a solution deployed through a IBM Cognos application.
* In this case, the case study was hospitalization risk for patients with juvenile diabetes. Like the congestive heart failure use case, this one used decision tree classification to create a risk model that would serve as the foundation for this application.
* The map gives an overview of hospitalization risk nationwide, with an interactive analysis of predicted risk by a variety of patient conditions and other characteristics.



* The figure above shows an interactive summary report of risk by patient population within a given node of the model, so that clinicians could understand the combination of conditions for this subgroup of patients.



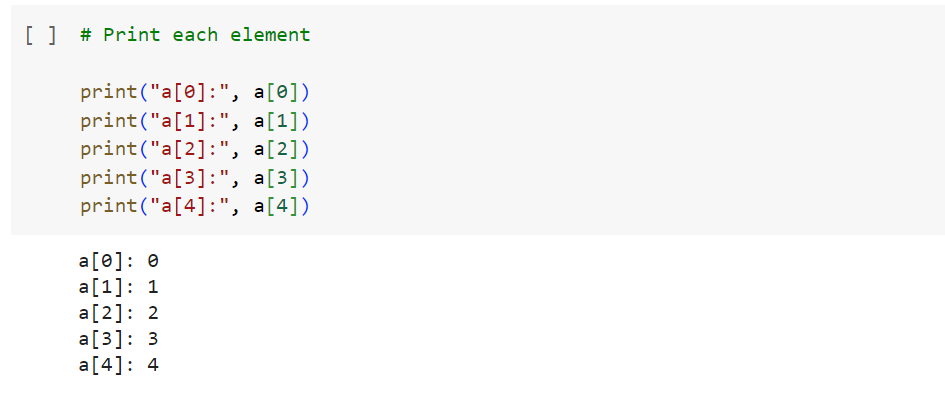
* And this report gives a detailed summary on an individual patient, including the patient's predicted risk and details about the clinical history, giving a concise summary for the doctor.

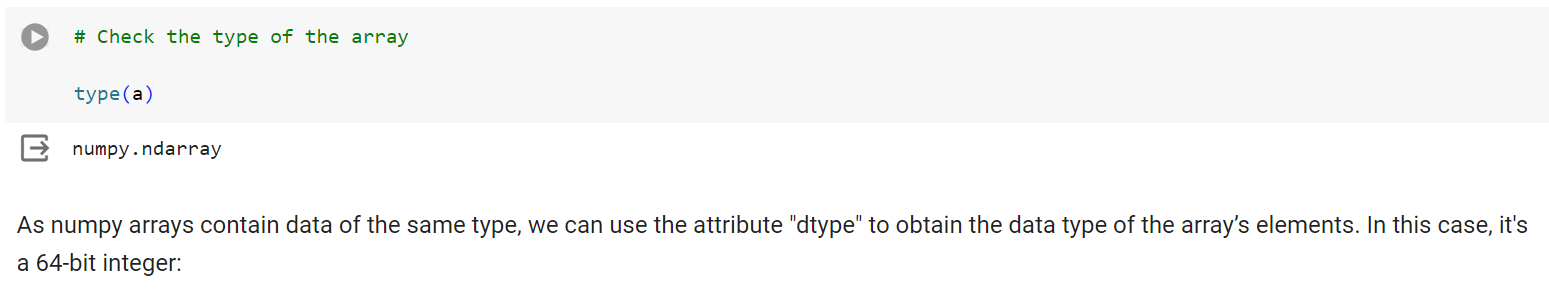
**Feedback: From Deployment to Feedback**

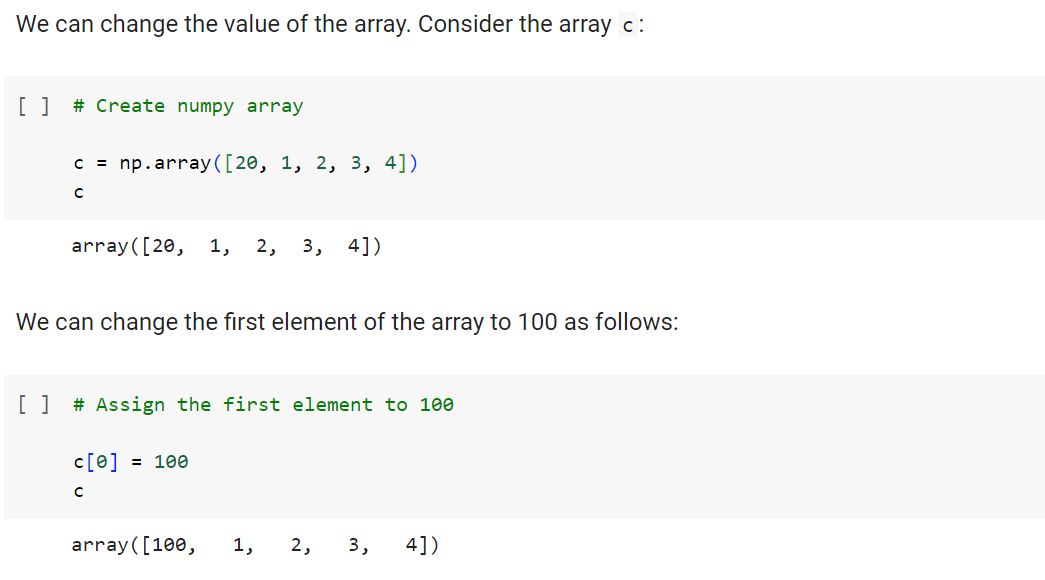
* Once in play, feedback from the users will help to refine the model and assess it for performance and impact.
* The value of the model will be dependent on successfully incorporating feedback and making adjustments for as long as the solution is required.
* So now, let's look at our case study again, to see how the Feedback portion of the methodology is applied.
* The plan for the feedback stage included these steps:
  + First, the review process would be defined and put into place, with overall responsibility for measuring the results of a "flying to risk" model of the congestive heart failure risk population. Clinical management executives would have overall responsibility for the review process.
  + Second, congestive heart failure patients receiving intervention would be tracked and their re-admission outcomes recorded.
  + Third, the intervention would then be measured to determine how effective it was in reducing re-admissions.
* For ethical reasons, congestive heart failure patients would not be split into controlled and treatment groups.
* Instead, readmission rates would be compared before and after the implementation of the model to measure its impact.
* After the deployment and feedback stages, the impact of the intervention program on re-admission rates would be reviewed after the first year of its implementation.
* Then the model would be refined, based on all of the data compiled after model implementation and the knowledge gained throughout these stages.
* Other refinements included: Incorporating information about participation
* in the intervention program, and possibly refining the model to incorporate
* detailed pharmaceutical data.
* If you recall, data collection was initially deferred because the pharmaceutical data was not readily available at the time.
* But after feedback and practical experience with the model, it might be determined that adding that data could be worth the investment of effort and time.
* We also have to allow for the possibility that other refinements might present themselves during the feedback stage.
* Also, the intervention actions and processes would be reviewed and very likely refined as well, based on the experience and knowledge gained through initial deployment and feedback.
* Finally, the refined model and intervention actions would be redeployed, with the feedback process continued throughout the life of the Intervention program.

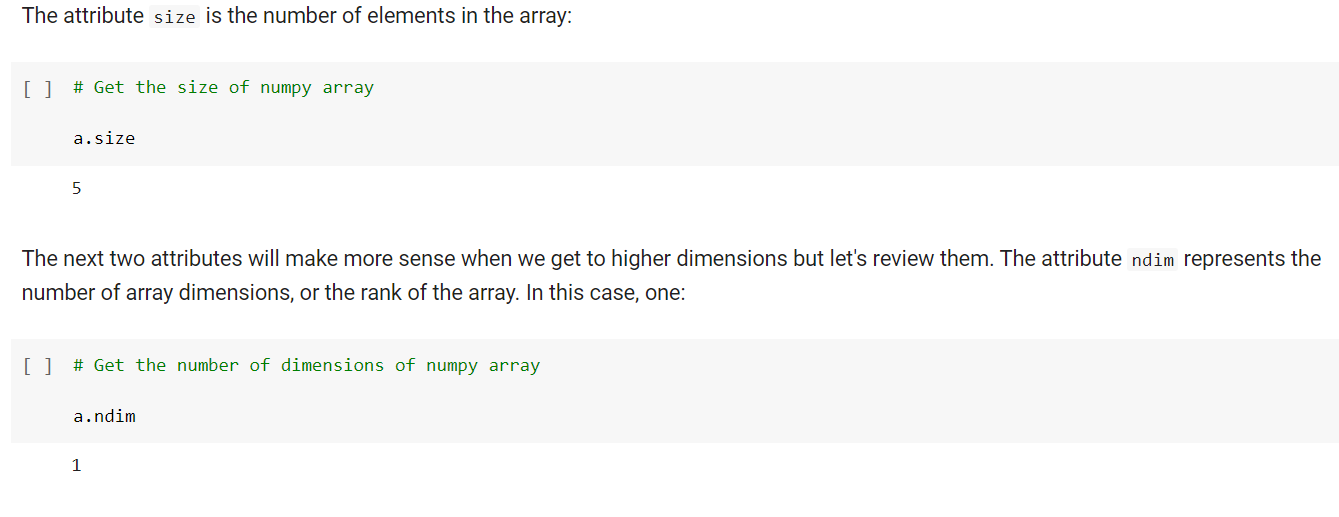
**Practical 2A**

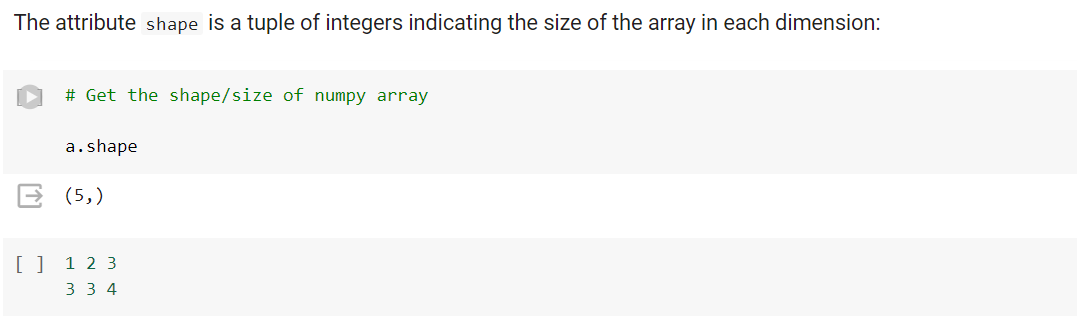
**Aim - Study Numpy library, Arrays, Dimensions- 2D, 3D, ND, Broadcasting, Indexing, Slicing Numpy Functions: array manipulation, string, arithmetic, statistical, Numpy Functions: arrange, linspace, random number generation, seed, reshape, ravel**

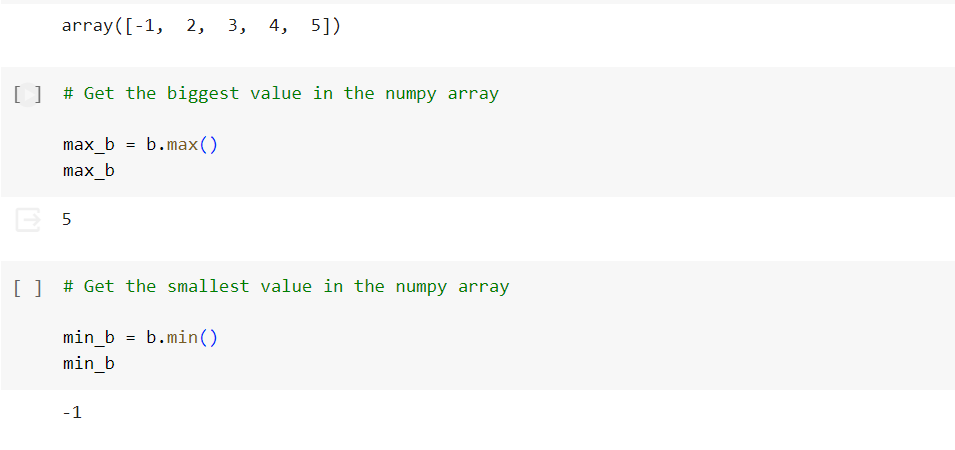
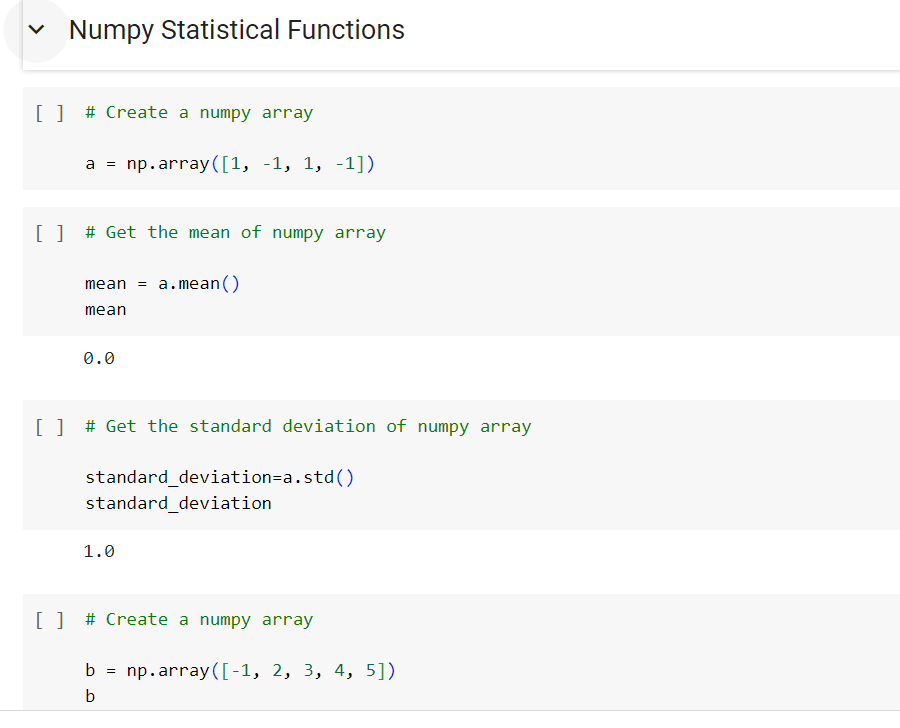


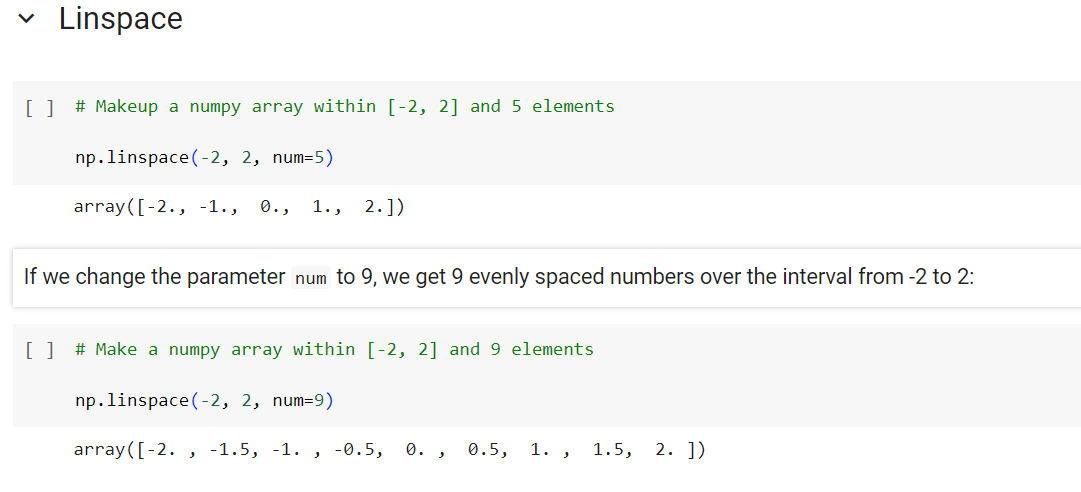
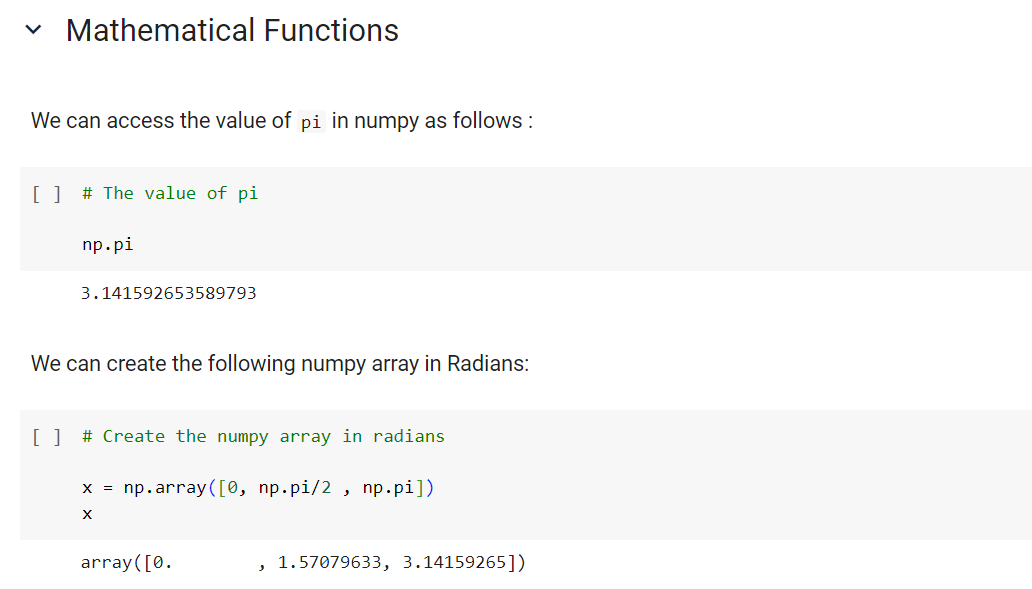


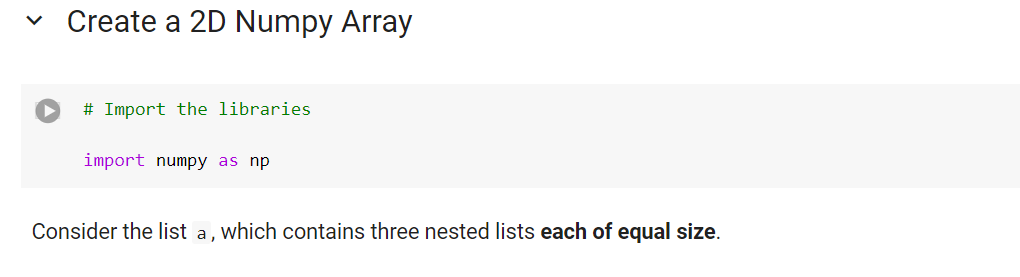


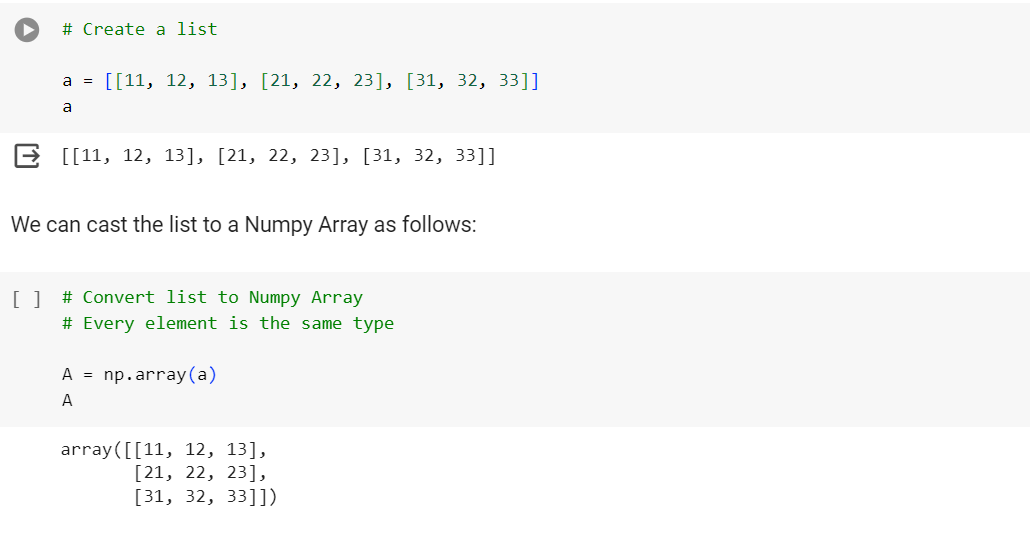


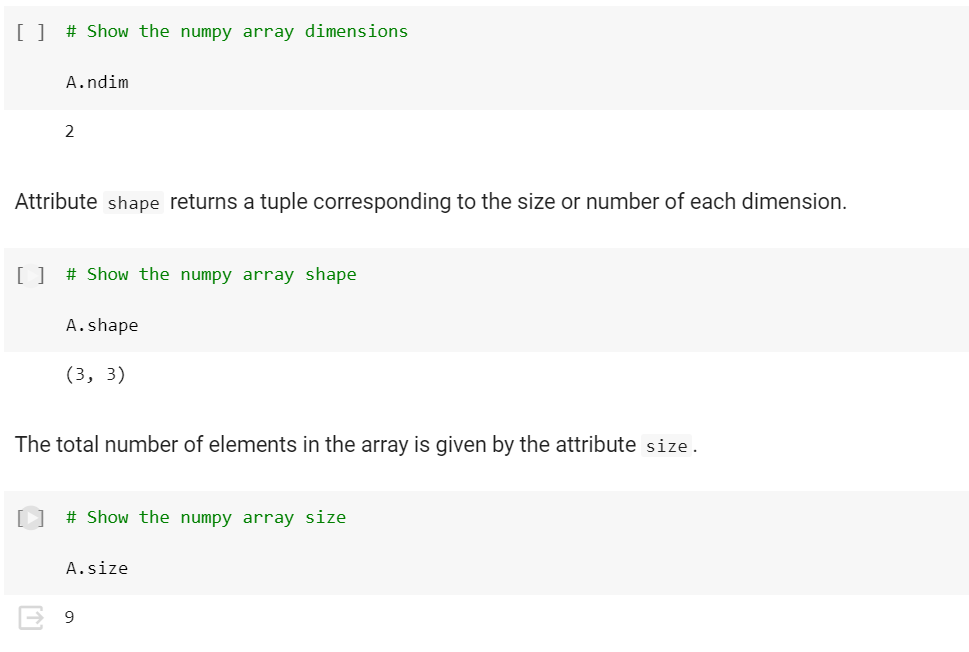


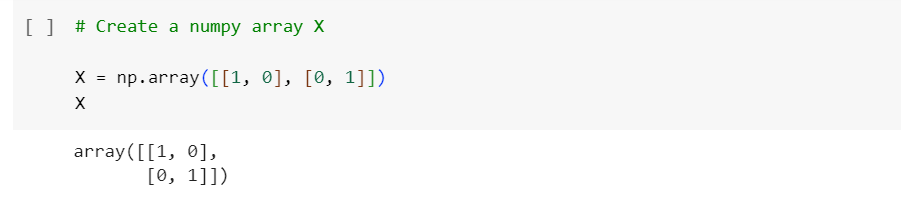


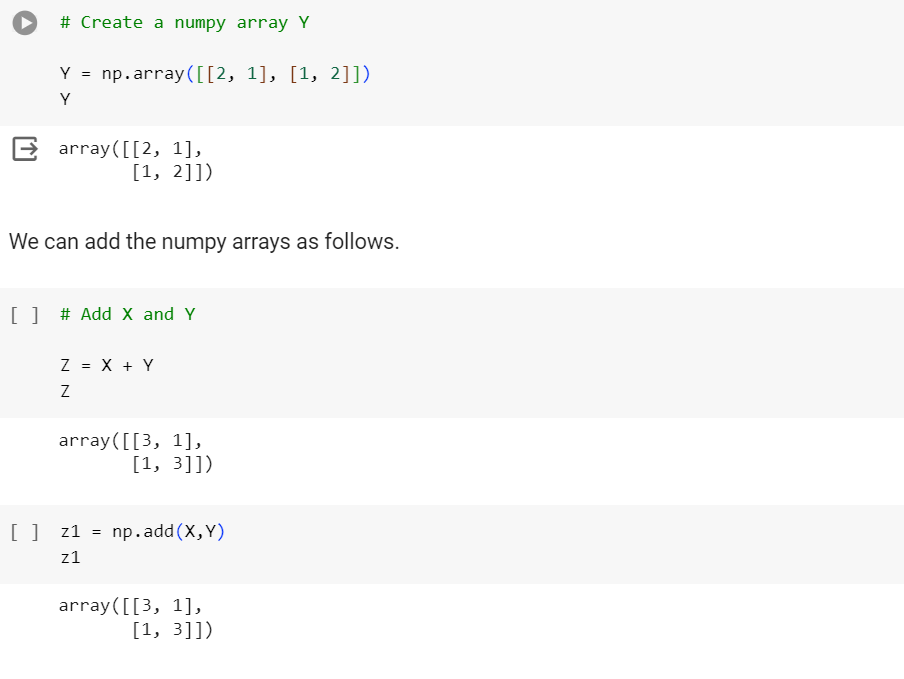


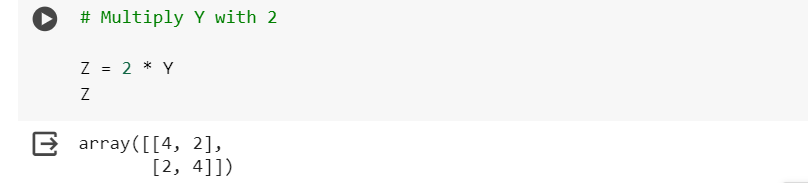


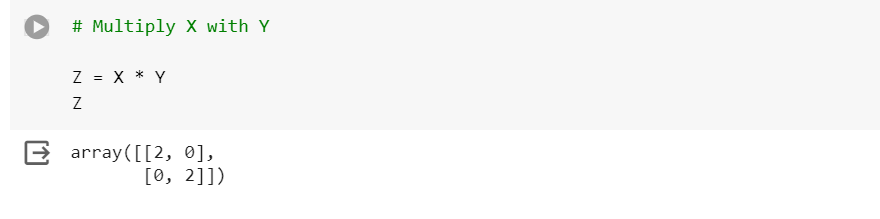


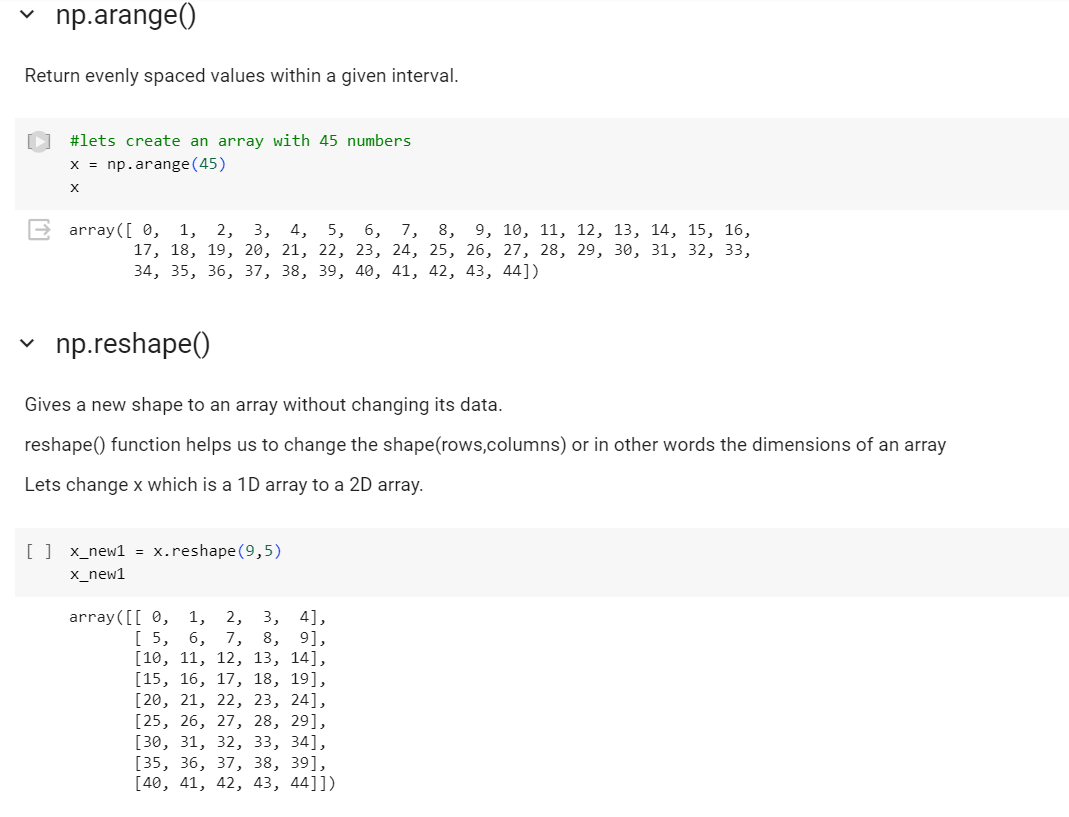








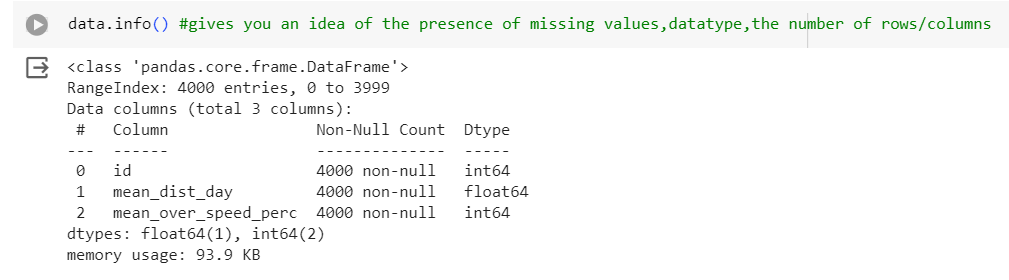
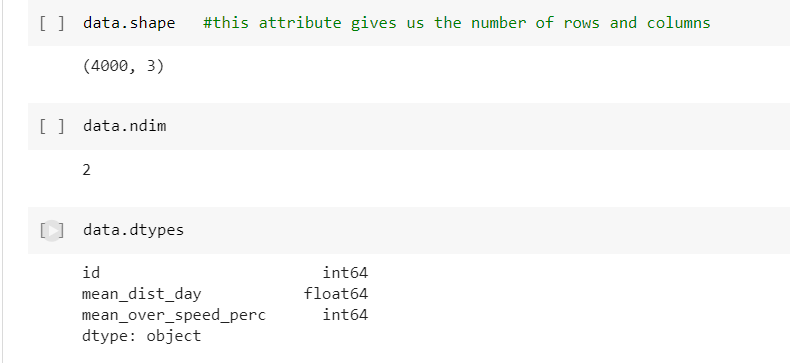
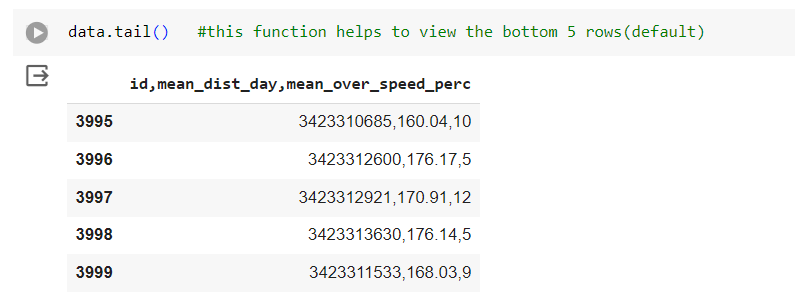
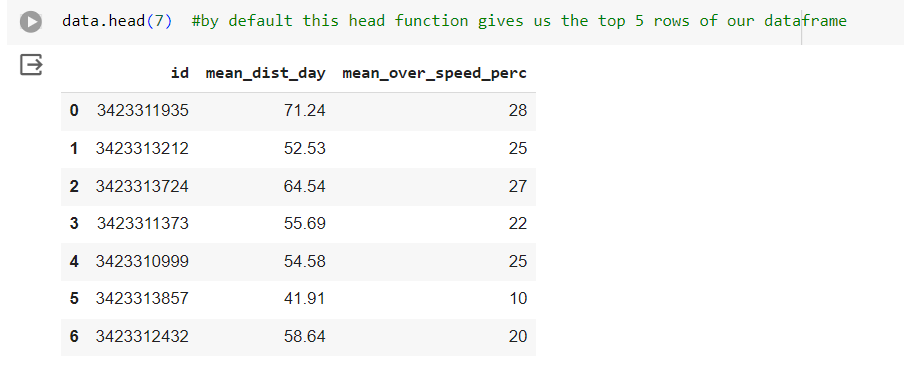
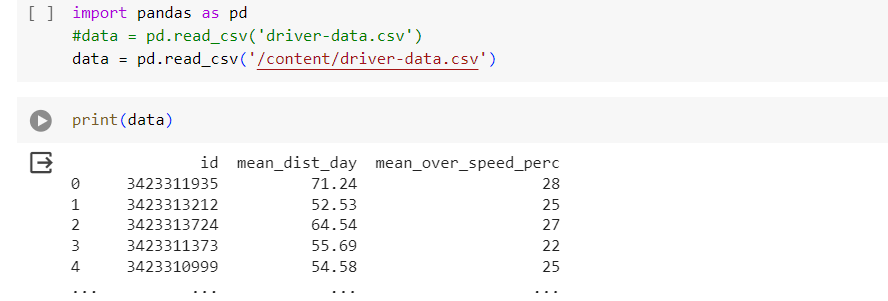


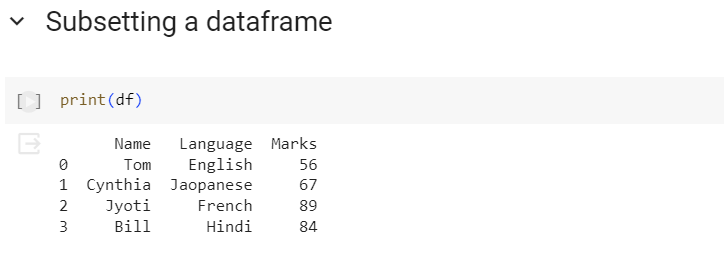
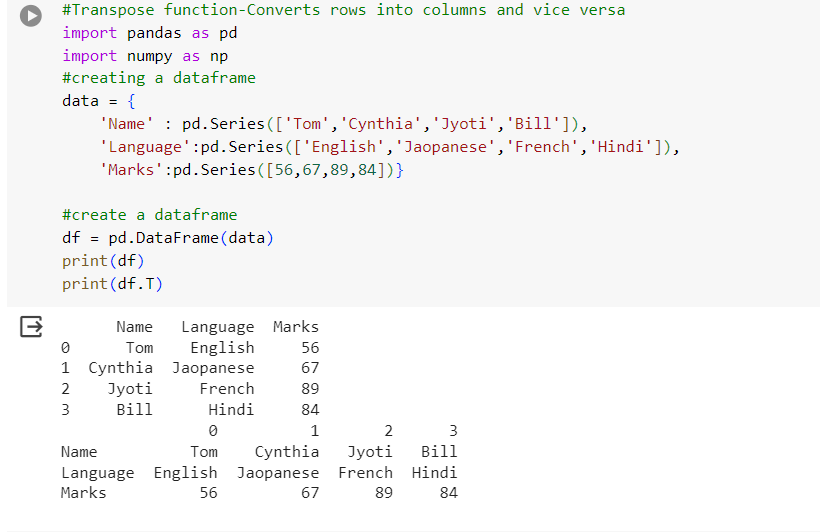


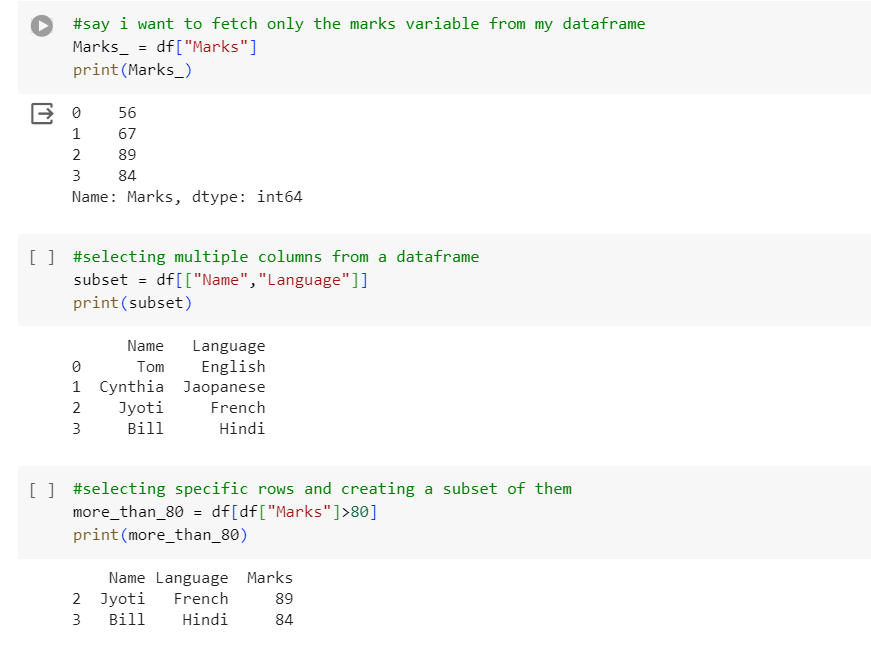
**Practical 2B**

**Aim - Study Pandas library, Series functions: empty, ndim, size, dtype, values, head, tail, DataFrame functions: datatype, transpose, empty, ndim, shape, size, values, head, tail,**

**DateTime**



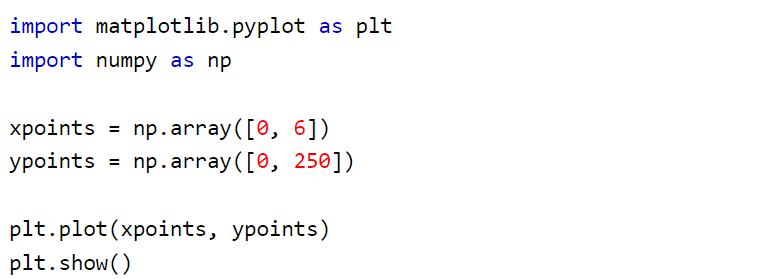


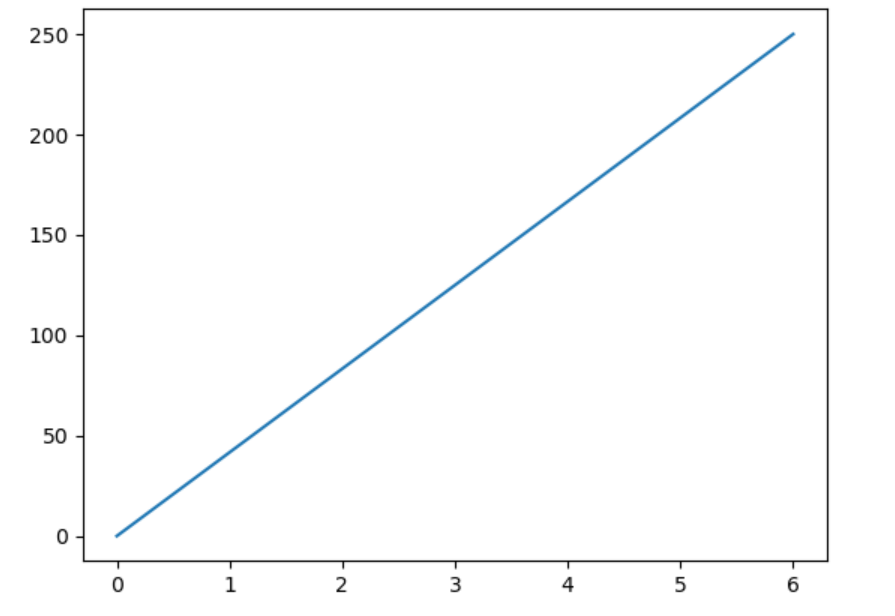


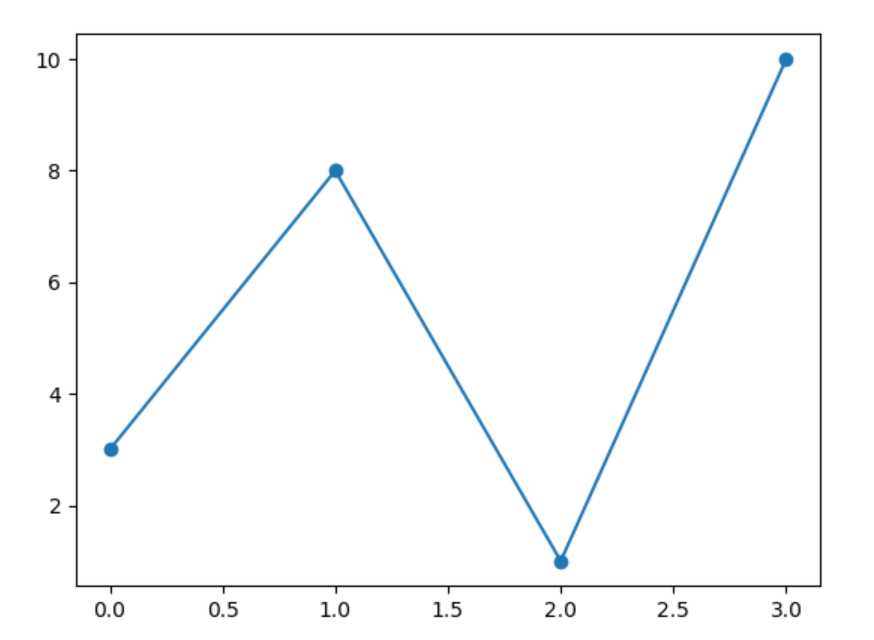
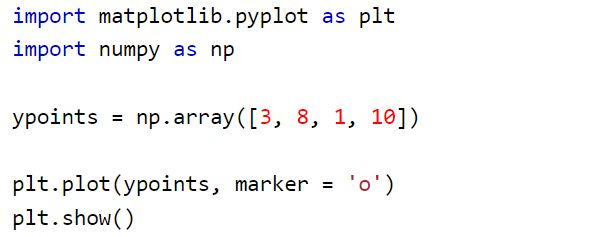
**Practical 2C**

**Aim - Study Matplotlib library, Seaborn, Plyplot, plotting, markers, line, labels, grid, subplot, scatter, bar, histogram, pie charts, countplot**

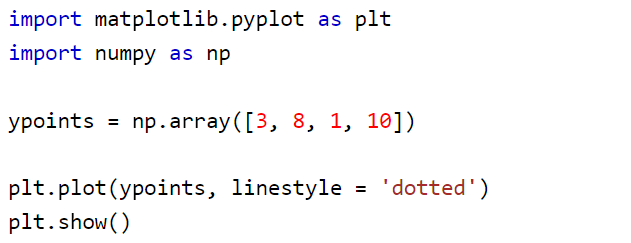
#plotting with matplotlibs pylot

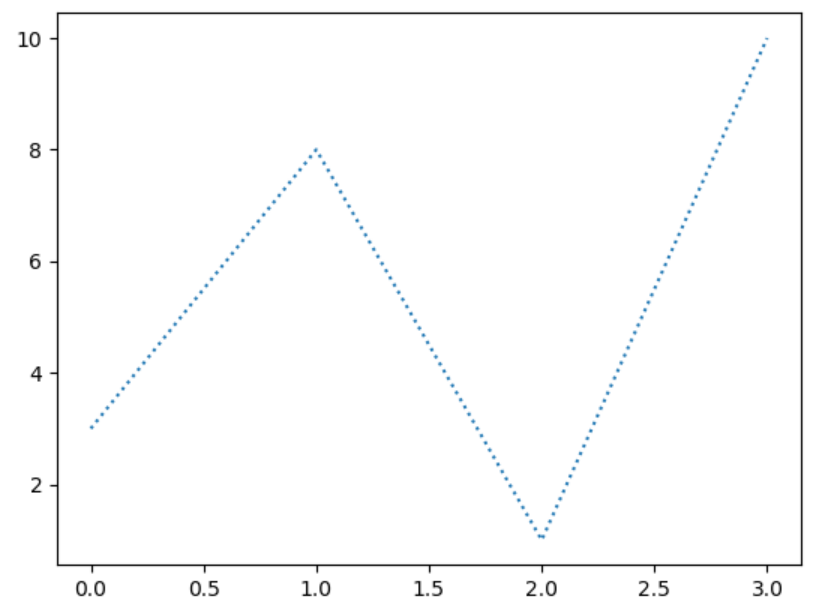


# Markers

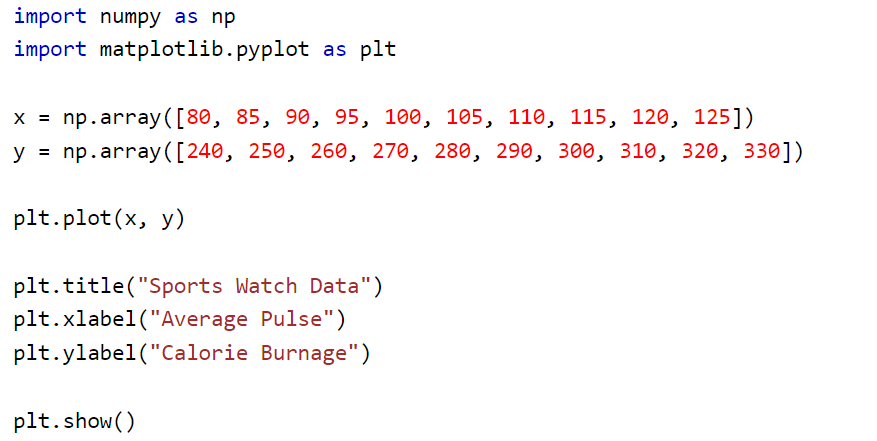


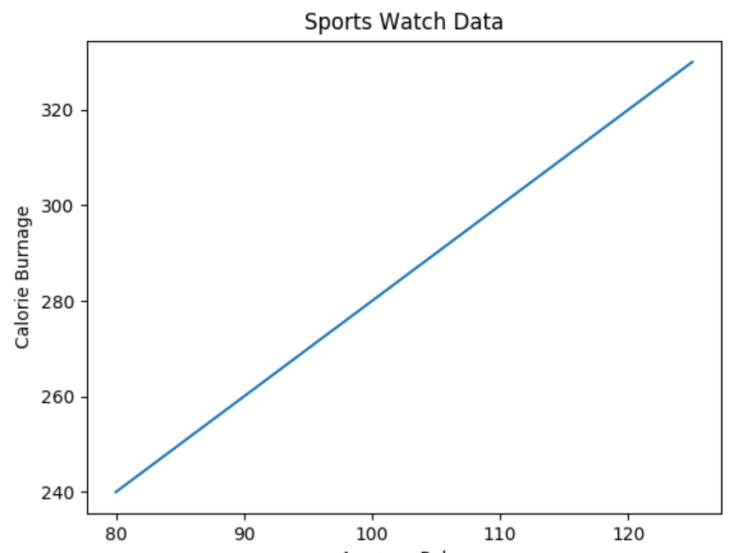
#Line type





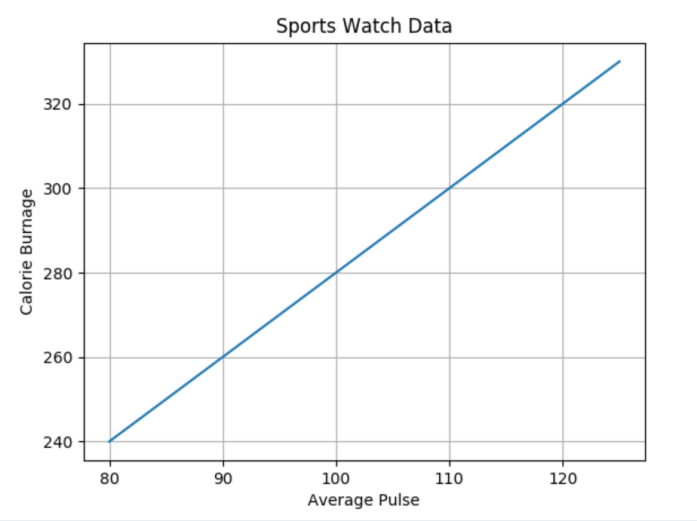
**# Matplotlib Labels and Title**



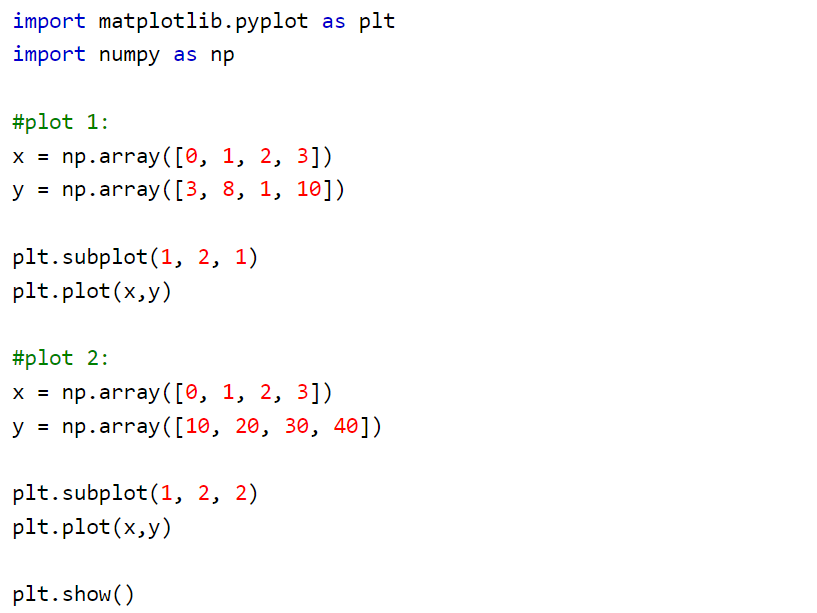


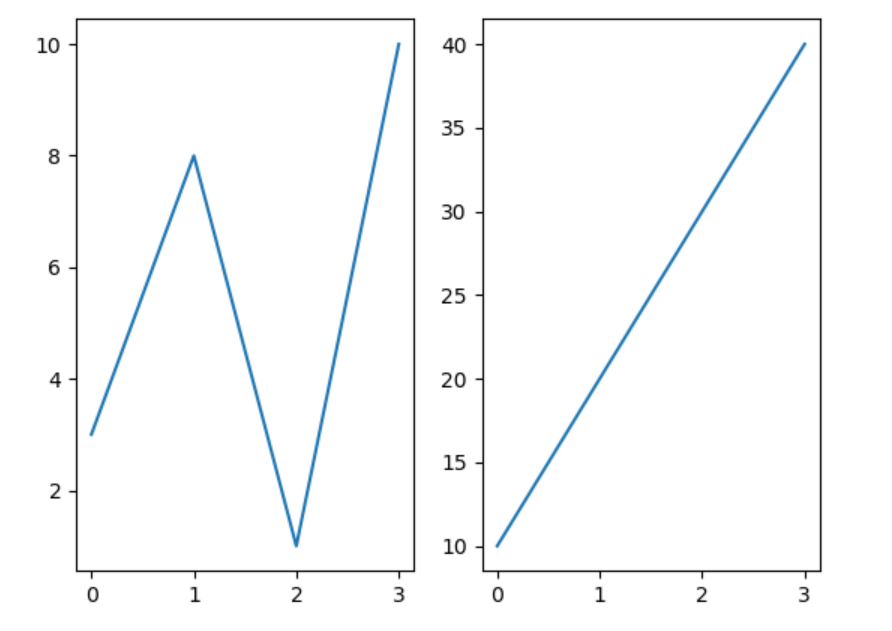
# Add Grid Lines to a Plot





# Matplotlib Subplot





|  |  |
| --- | --- |
| # Scatter Plot |  |
| import matplotlib.pyplot as plt import numpy as np  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6]) y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  plt.scatter(x, y) plt.show() |  |
| # Bar Plot |  |
| import matplotlib.pyplot as plt import numpy as np  x = np.array(["A", "B", "C", "D"]) y = np.array([3, 8, 1, 10])  plt.bar(x,y) plt.show() |  |
| # Histogram Plot |  |
| import matplotlib.pyplot as plt import numpy as np  x = np.random.normal(170, 10, 250)  plt.hist(x) plt.show() |  |
| # Pie Chart |  |
| import matplotlib.pyplot as plt import numpy as np  y = np.array([35, 25, 25, 15])  plt.pie(y) plt.show() |  |
|  |  |

**Practical 3A**

**Aim - Study Data Analysis, Import Libraries & datasets, load data, check data type, shape, drop columns, merge datasets, sort, concatenate, statistical summary of data, skewness, co-relation**

**Import Libraries**

import pandas as pd

import numpy as np

**Import datasets**

dataset = pd.read\_csv("/content/melb\_data.csv")

**Load Data**

df = pd.DataFrame(dataset)

**check data type**

type(df)

**Output:**

**pandas.core.frame.DataFrame**

**Shape**

df.shape

**(13580, 21)**

**drop columns**

dataset\_2 = df.drop(['Rooms'], axis=1)

dataset\_2.shape

**(13580, 20)**

**merge datasets**

dataset\_2 = pd.read\_excel('/content/automobile.xlsx')

combined\_data = pd.merge(df, dataset\_2, on='Address')

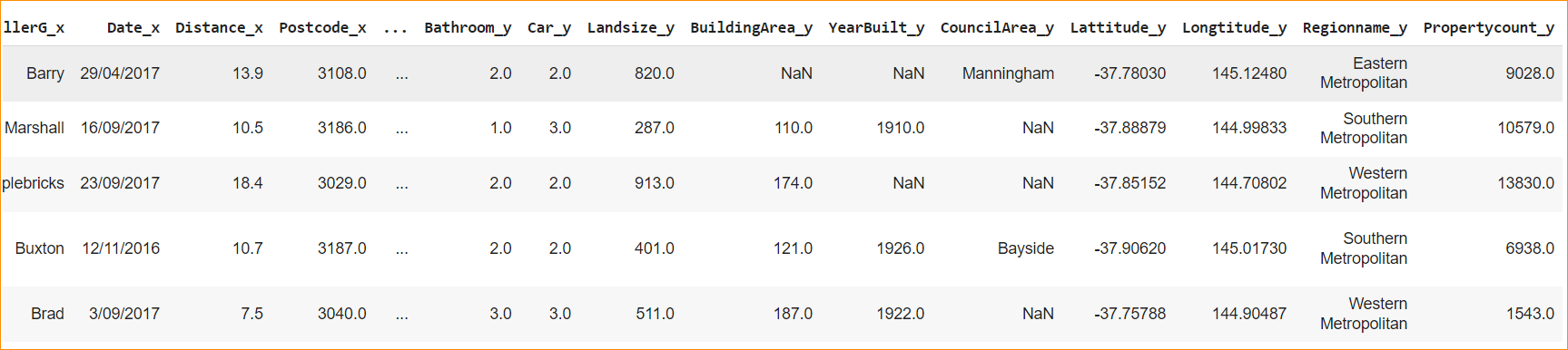
combined\_data.shape

**(14002, 41)**

**Sort**

dataset\_3 = combined\_data.sort\_values(by=['Address'])

dataset\_3.head()



**Concatenate**

final\_data = pd.concat([combined\_data, dataset\_3])

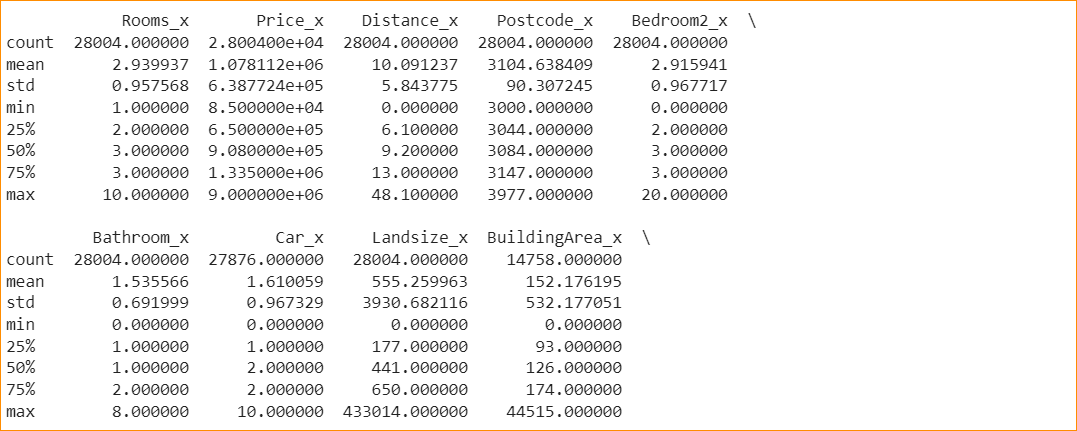
final\_data.shape

**(28004, 41)**

**Statistical summary of data**

summary = final\_data.describe()

print(summary)

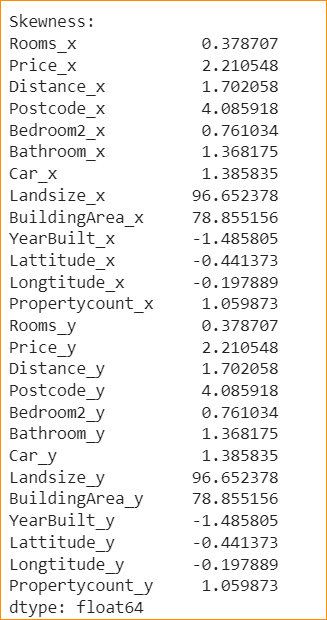


**Skewness**

skewness = final\_data.skew()

print("Skewness:")

print(skewness)

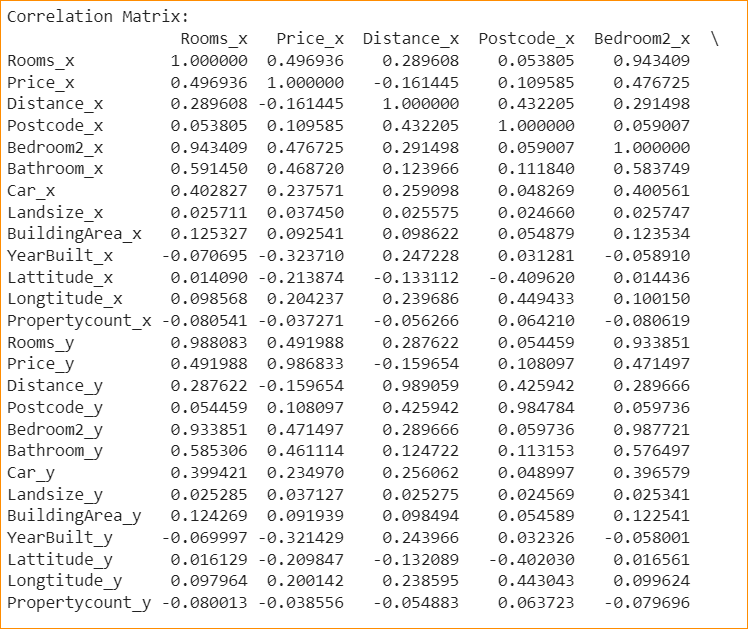


**co-relation**

correlation\_matrix = final\_data.corr()

print("Correlation Matrix:")

print(correlation\_matrix)



**Practical 3B**

**Aim - Study Data Wrangling, Pre-processing Data - Dealing with Missing values, Correcting Data Format, Data standardization, Data Normalization, Binning, Turning categorical variables into quantitative variables**

**Dealing with Missing values**

data\_without\_missing\_rows = df.dropna()

# Remove columns with any missing values

data\_without\_missing\_columns = df.dropna(axis=1)

data\_without\_missing\_columns.shape

(13580, 17)

df.replace(999, 0, inplace=True)

**Correcting Data Format**

df['Rooms'] = pd.to\_numeric(df['Rooms'])

df.describe()



**Data Normalization**

**Simple Feature Scaling**

df['Price\_scaled'] = df['Price'] / df['Price'].max()

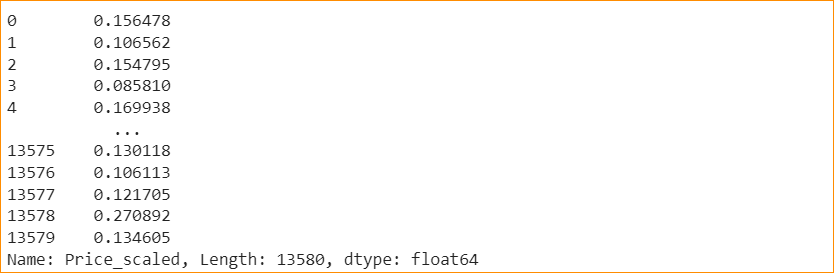
print(df['Price\_scaled'])



**Min-Max Scaling in Python**

df['Price\_scaled'] = (df['Price'] - df['Price'].min()) / (df['Price'].max() - df['Price'].min())

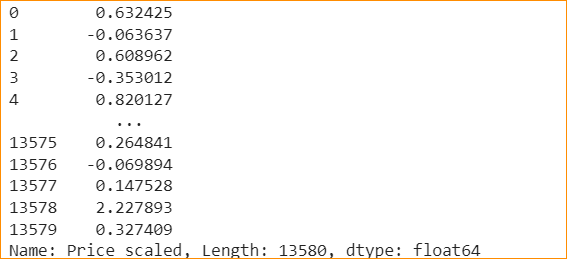
print(df['Price\_scaled'])



**Z-Score in Python**

df['Price\_scaled'] = (df['Price'] - df['Price'].mean()) /df['Price'].std()

print(df['Price\_scaled'])



**Binning**

bins = np.linspace(min(df['Price']), max(df['Price']), num=4)

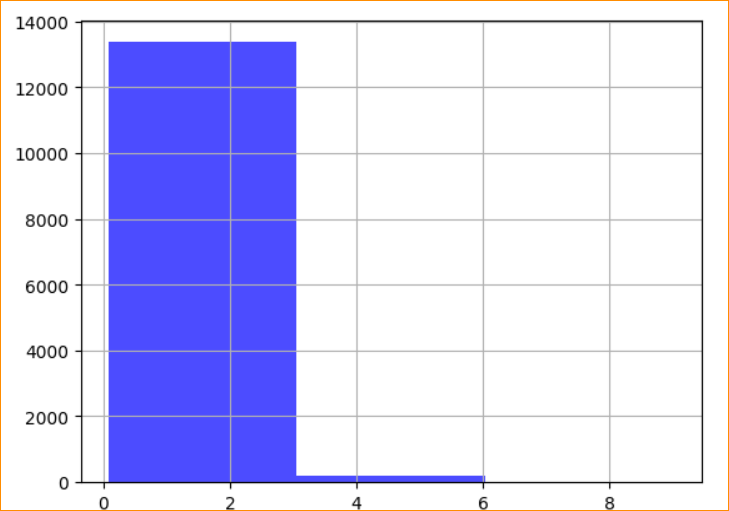
group\_names = ['Low', 'Medium', 'High']

df['price\_binned'] = pd.cut(df['Price'], bins, labels=group\_names, include\_lowest=True)

import matplotlib.pyplot as plt

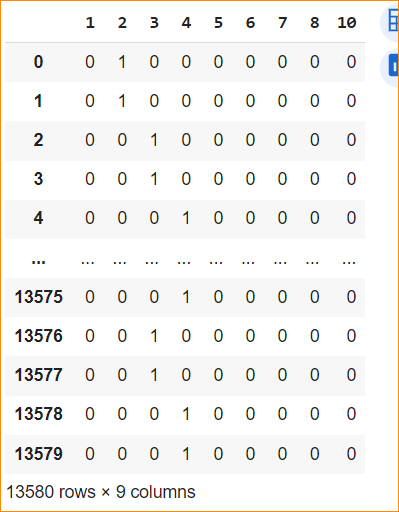
df['Price'].hist(bins=3, color='blue', alpha=0.7)

plt.grid(True)



**Turning categorical variables into quantitative variables**

pd.get\_dummies(df["Rooms"])



**Practical 4**

**Aim - Study Exploratory Data Analysis, Descriptive Statistics, Categorical Variables - box plots, Value Counts, Scatterplot , Group By, Pivot Table, Heat Map**

import pandas as pd

import numpy as np

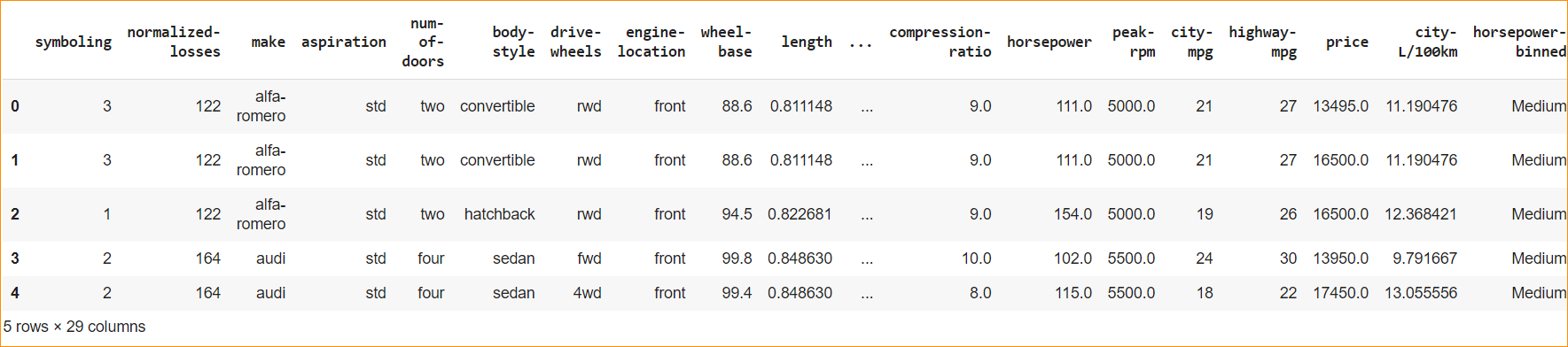
import matplotlib.pyplot as plt

import scipy.stats as stats

import seaborn as sb

df = pd.read\_csv('/content/automobile.csv')

df.head()



**Value Counts**

df['drive-wheels'].value\_counts()

fwd 118

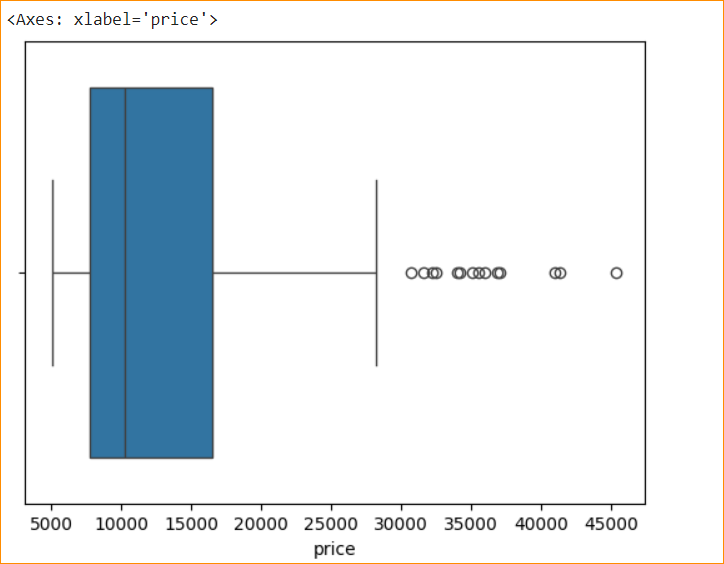
rwd 75

4wd 8

Name: drive-wheels, dtype: int64

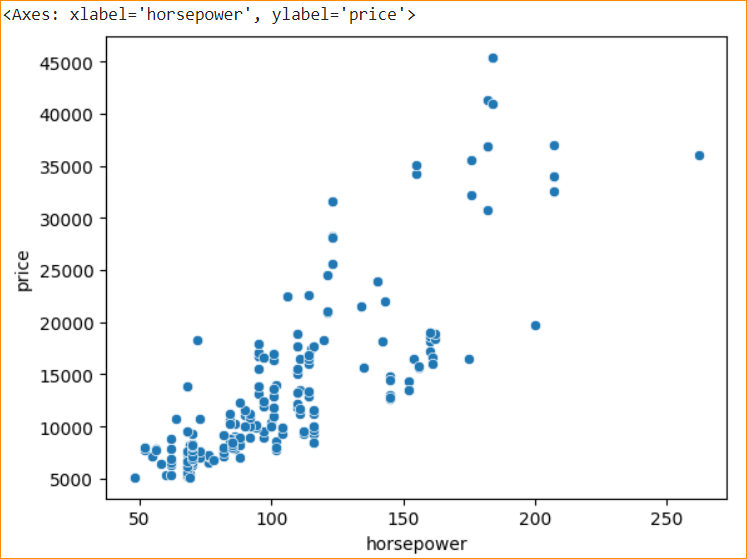
**Box Plot**

sb.boxplot(x = df['price'])

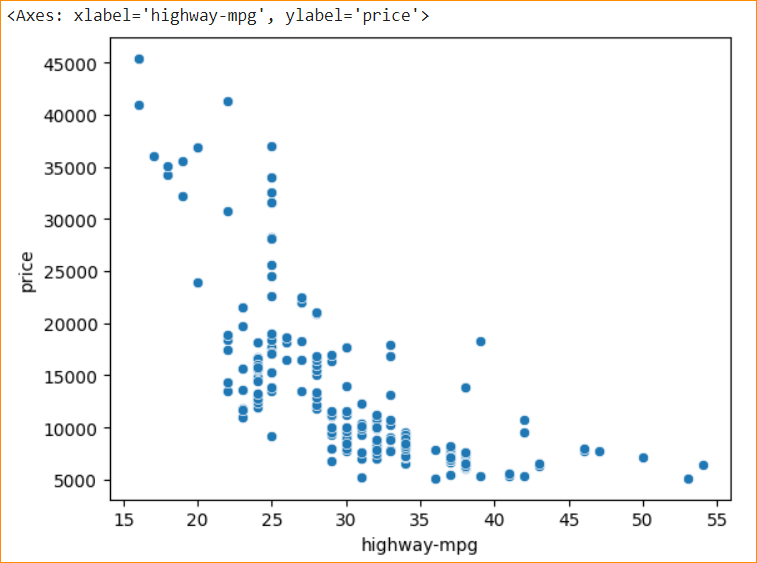


**Scatterplot**

sb.scatterplot(x = df['horsepower'], y = df['price'])



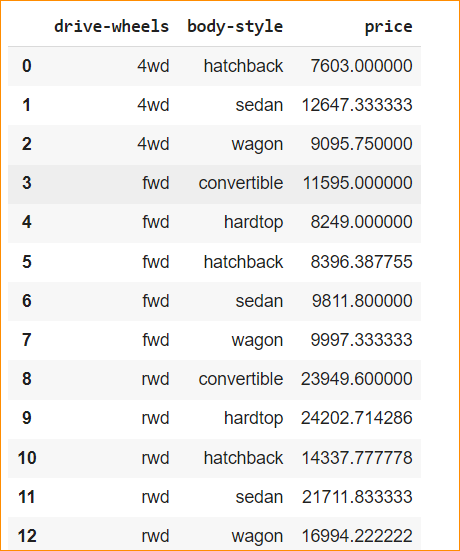
sb.scatterplot(x = df['highway-mpg'], y = df['price'])



Group By

df\_grp = df\_test.groupby(['drive-wheels', 'body-style'], as\_index = False).mean()

df\_grp



Pivot Table

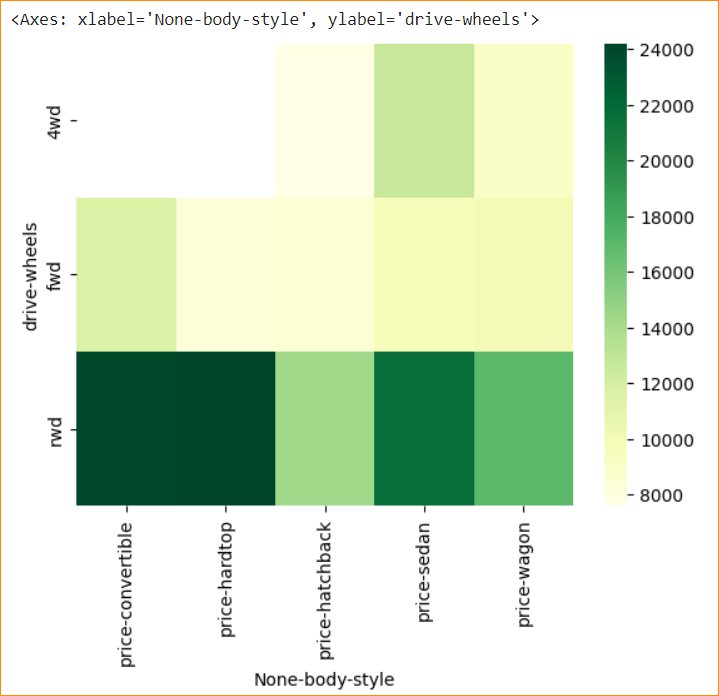
df\_pivot = df\_grp.pivot(index = 'drive-wheels', columns = 'body-style')

df\_pivot



Heat Map

sb.heatmap(df\_pivot, cmap= 'YlGn')



**Practical 5**

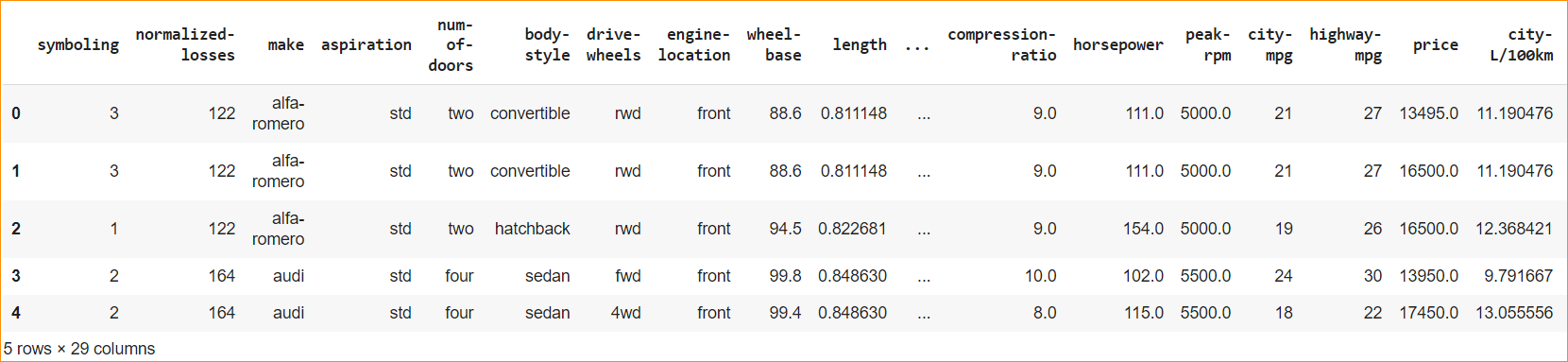
**Aim - Study Model Selection, Model Evaluation using Visualization, Measures for In-Sample Evaluation**

import pandas as pd

import numpy as np

df = pd.read\_csv('/content/automobile.csv')

df.head()



**LinearRegression**

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

x = df[['highway-mpg']]

y = df['price']

lm.fit(x,y)

lm\_for\_copy = LinearRegression()

x1 = df\_copy[['highway-mpg']]

y1 = df\_copy['price']

lm\_for\_copy.fit(x1,y1)

lm\_for\_copy.predict(x)

intercept = lm\_for\_copy.intercept\_

slope = lm\_for\_copy.coef\_

print(intercept)

print(slope)

#Predicting a value of dependent variable

desired\_mpg = int(input('Enter a highway\_mpg for predicting price: '))

y = slope \* desired\_mpg + intercept

print(y)

38466.36687902652

[-822.72963836]

Enter a highway\_mpg for predicting price: 27

[16252.66664334]

**Multiple Regression**

multi\_reg\_model = LinearRegression()

X = df[['highway-mpg', 'horsepower']]

Y = df['price']

multi\_reg\_model.fit(X,Y)

multi\_reg\_model.predict(X)

intercept = multi\_reg\_model.intercept\_

slope = multi\_reg\_model.coef\_

print(intercept)

print(slope)

desired\_mpg = int(input('Enter a highway\_mpg for predicting price: '))

horsepower = int(input('Enter a horsepower for predicting price: '))

y = (slope[0] \* desired\_mpg + intercept) + (slope[1] \* horsepower + intercept)

print(y)

3489.3548324887633

[-176.3352949 146.30647722]

Enter a highway\_mpg for predicting price: 20

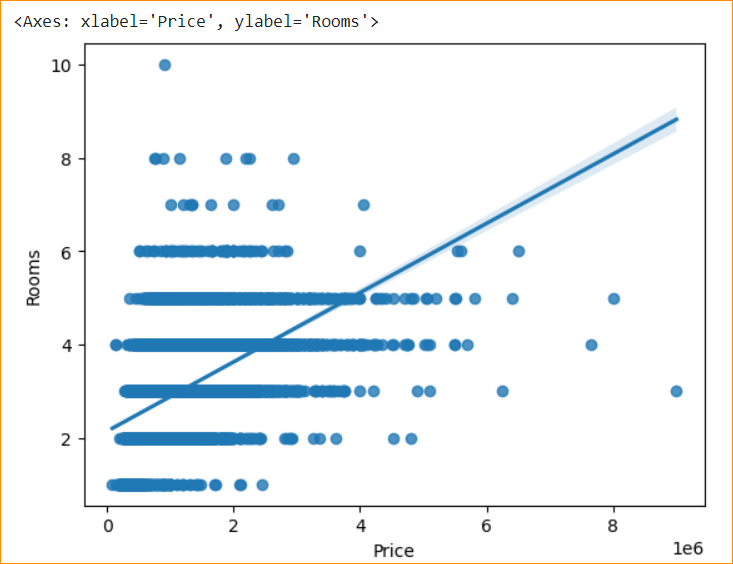
Enter a horsepower for predicting price: 1000

149758.4809892978

**Regression Plot**

import seaborn as sns

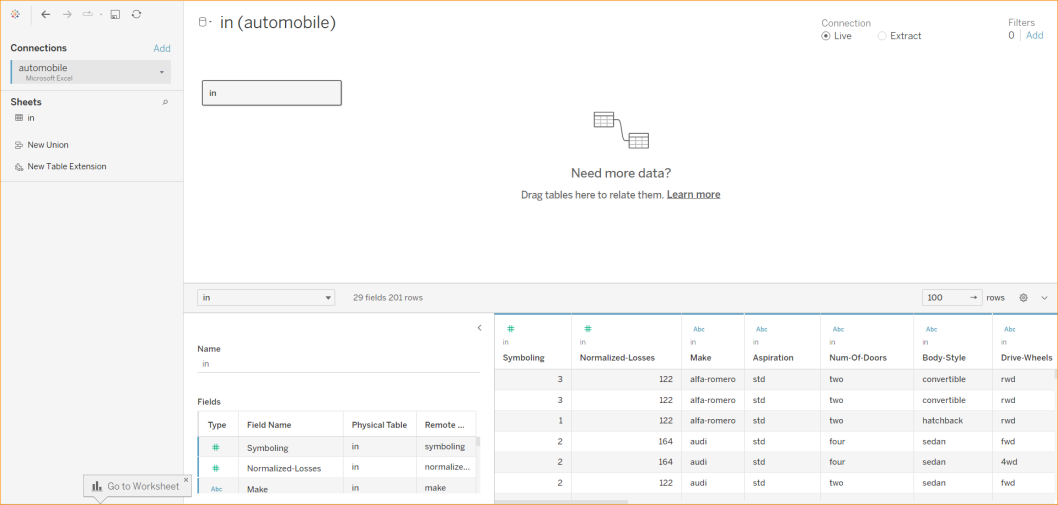
sns.regplot(x="Price",y="Rooms",data=df)



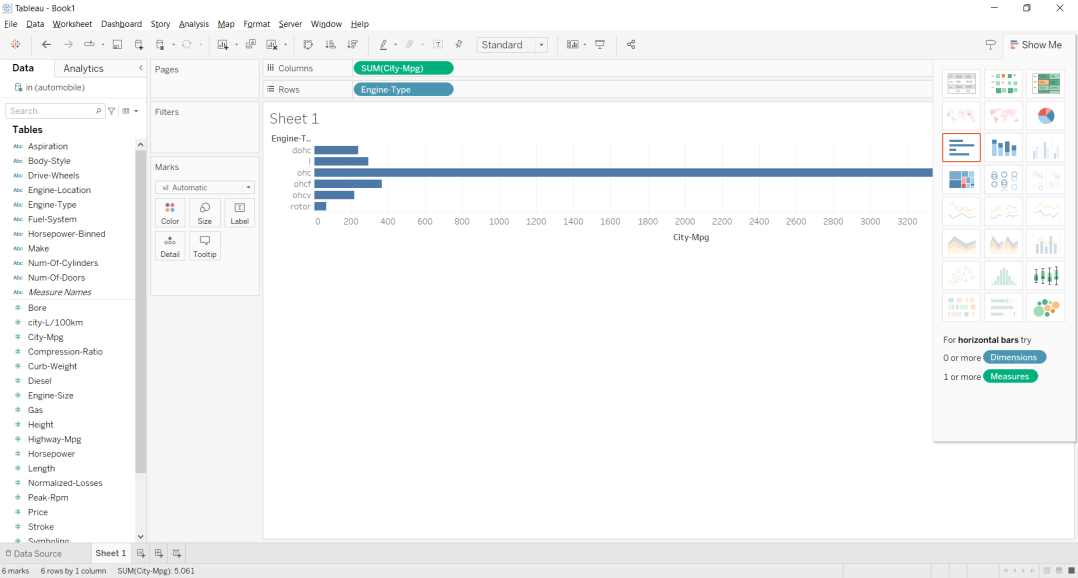
**Practical 6**

**Aim - Implement Data Visualization using Tableau**

**Upload Excel Data**



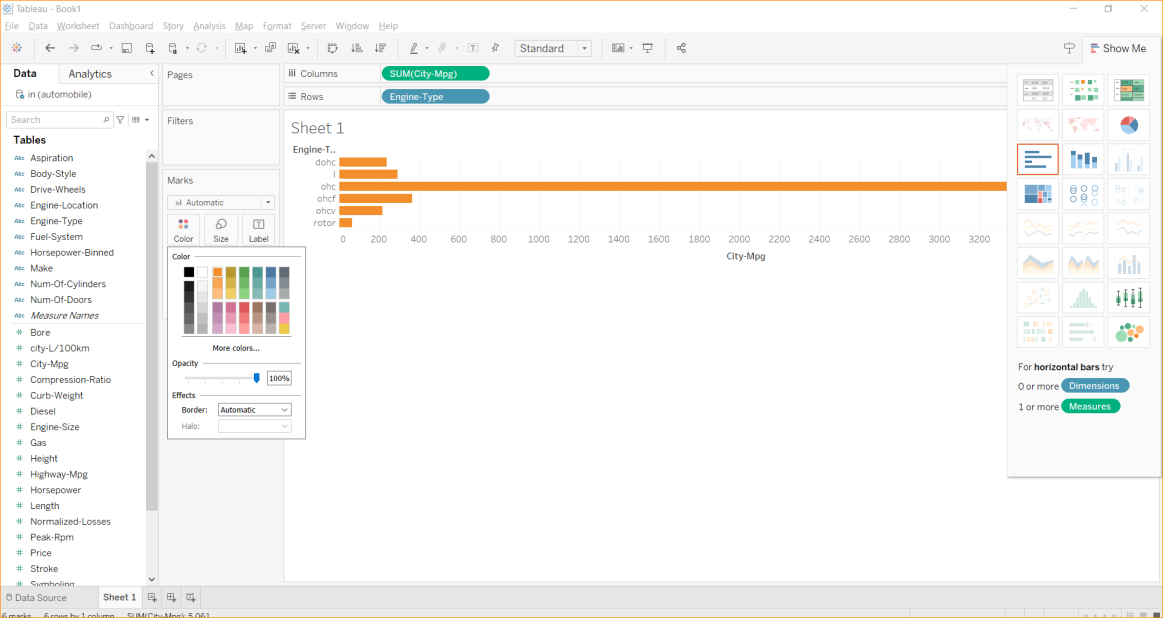
**Draw Horizontal Bar**



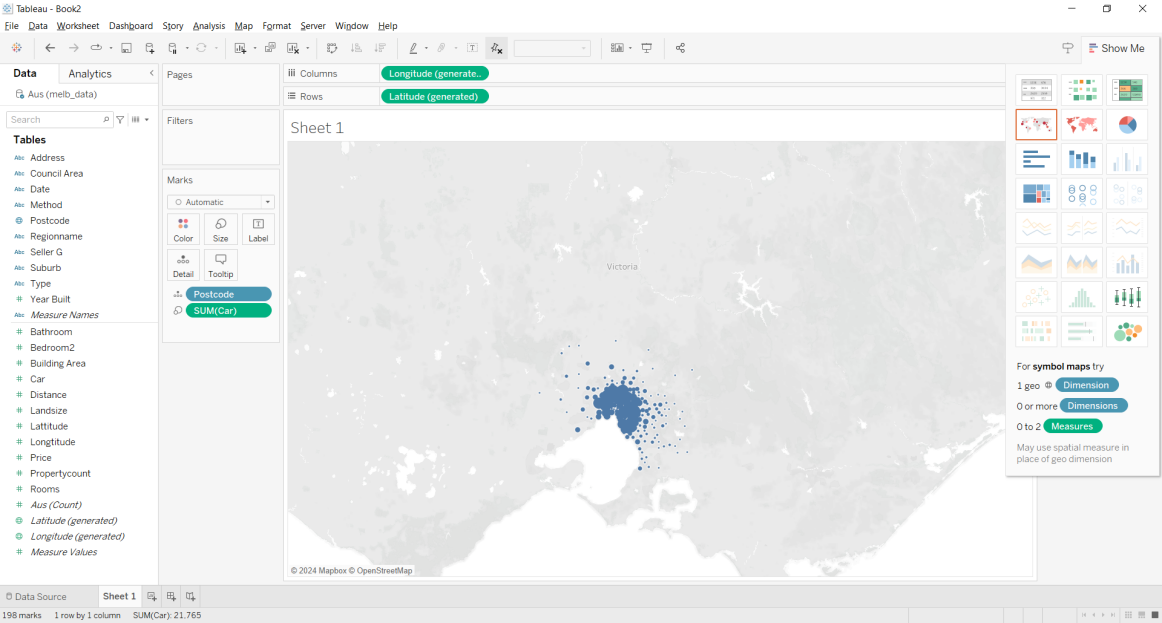
**Draw Stack Bar**



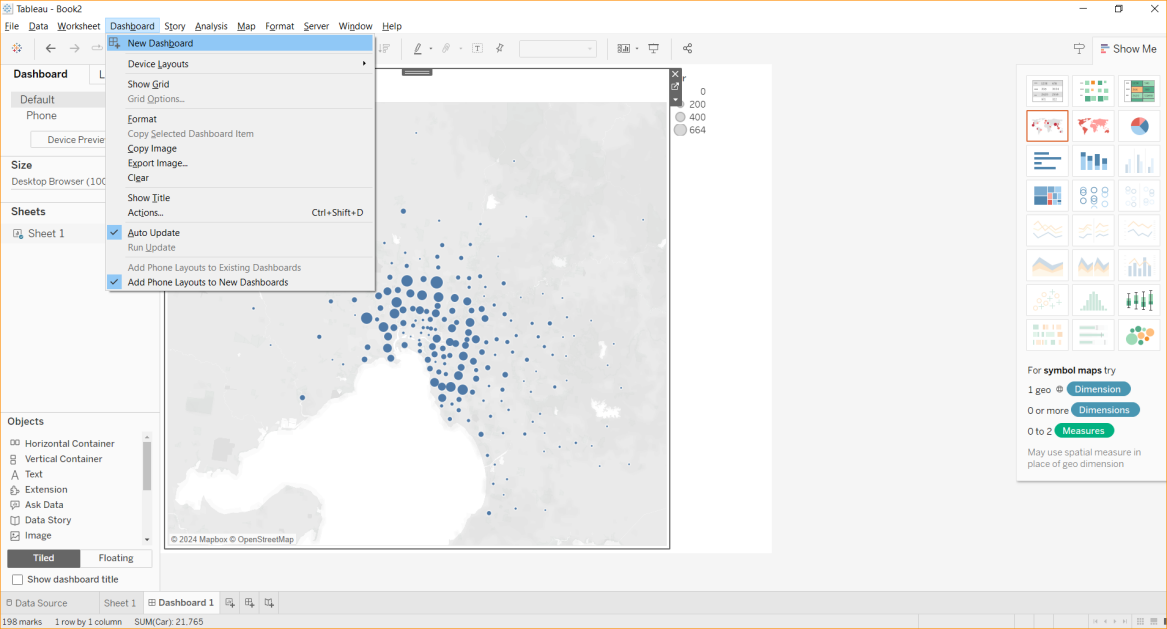
**Change Color**



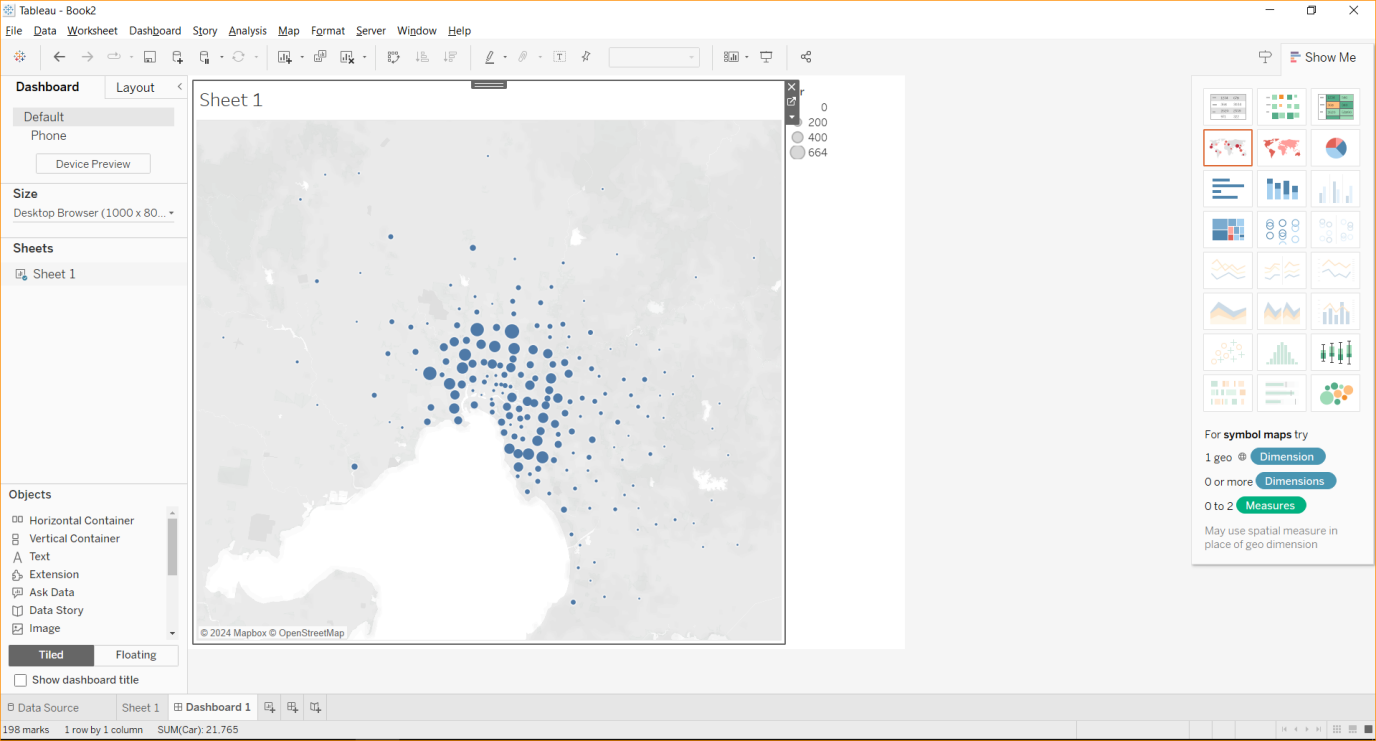
**Symbols Map**



**Create New Dashboard**



**Add Sheet To dashboard**

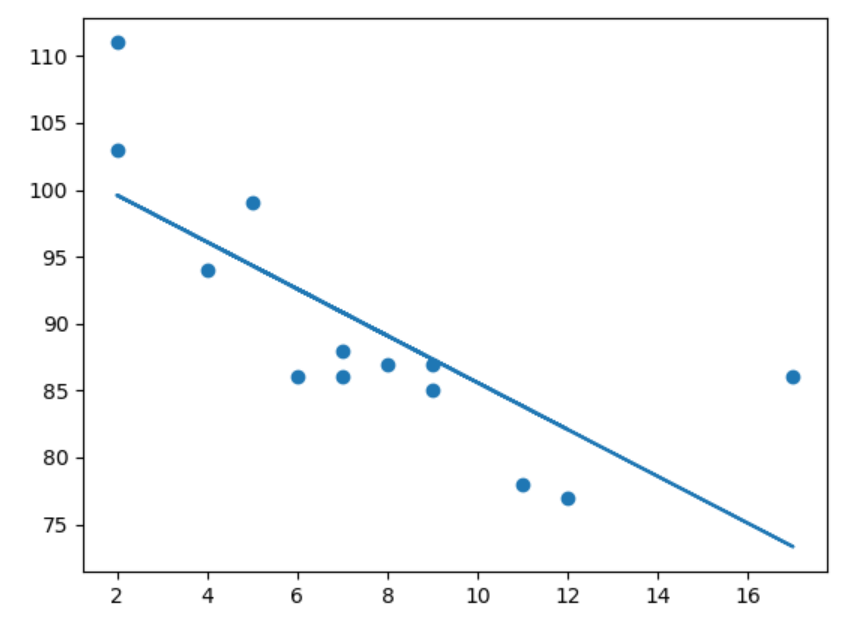


**Practical 7**

**Aim - Implement Regression – Linear, Logistic**

**# Linear Regression**

import matplotlib.pyplot as plt  
from scipy import stats  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
def myfunc(x):  
  return slope \* x + intercept  
  
mymodel = list(map(myfunc, x))  
  
plt.scatter(x, y)  
plt.plot(x, mymodel)  
plt.show()



# Predict Future Values

# Example: Let us try to predict the speed of a 10 years old car.

speed = myfunc(10)  
print(speed)

Output: 85.59308314937454

# Logistic Regression

import numpy  
from sklearn import linear\_model  
  
#X represents the size of a tumor in centimeters.

X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1,1)

#Note: X has to be reshaped into a column from a row for the LogisticRegression() function to work.

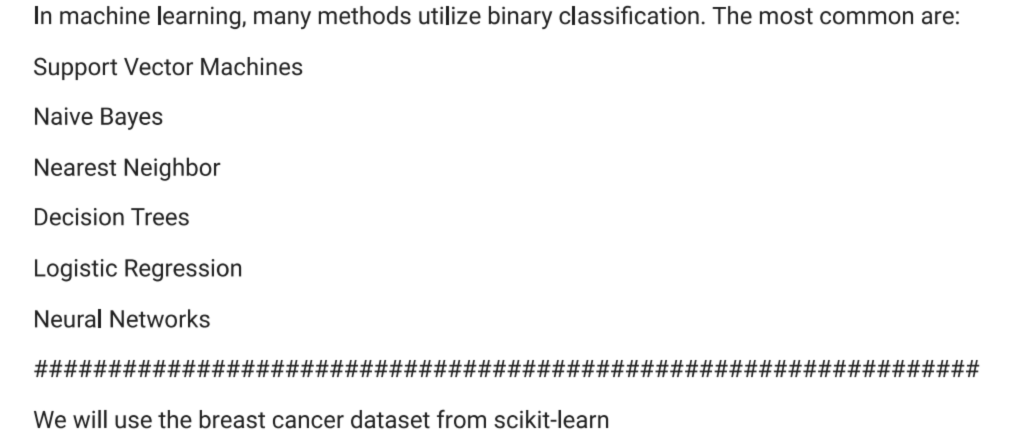
#y represents whether or not the tumor is cancerous (0 for "No", 1 for "Yes").

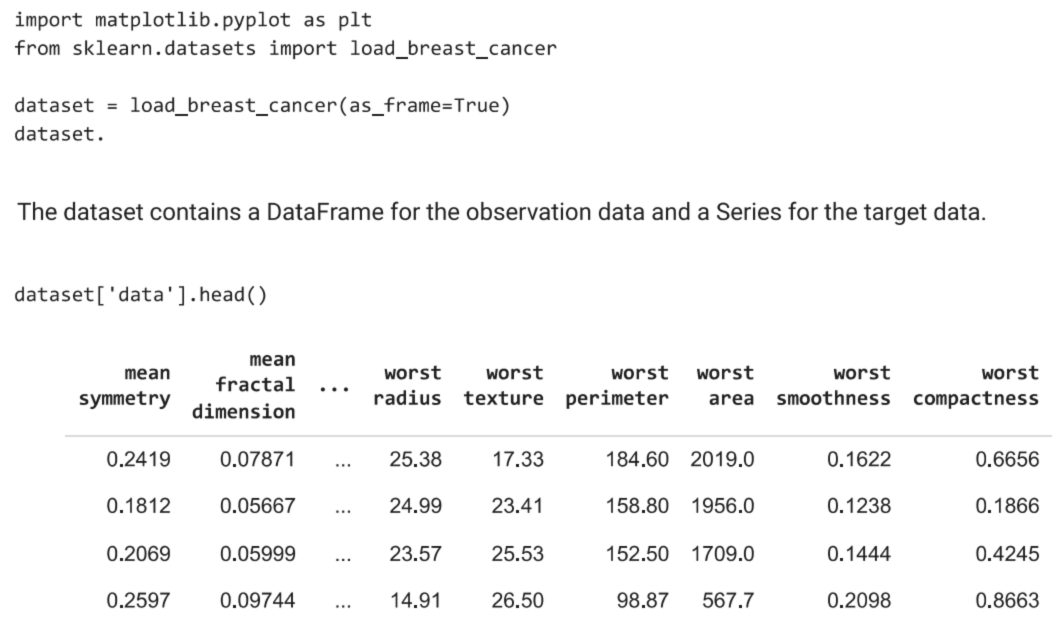
y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])  
  
logr = linear\_model.LogisticRegression()  
logr.fit(X,y)  
  
#predict if tumor is cancerous where the size is 3.46mm:  
predicted = logr.predict(numpy.array([3.46]).reshape(-1,1))  
print(predicted)

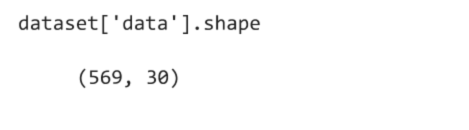
Outut: [0]

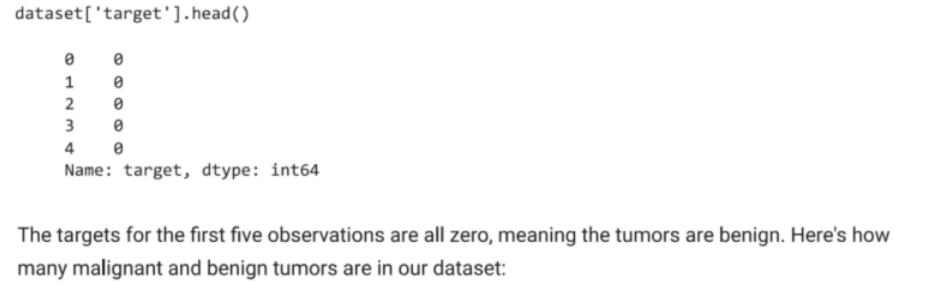
**Practical 8**

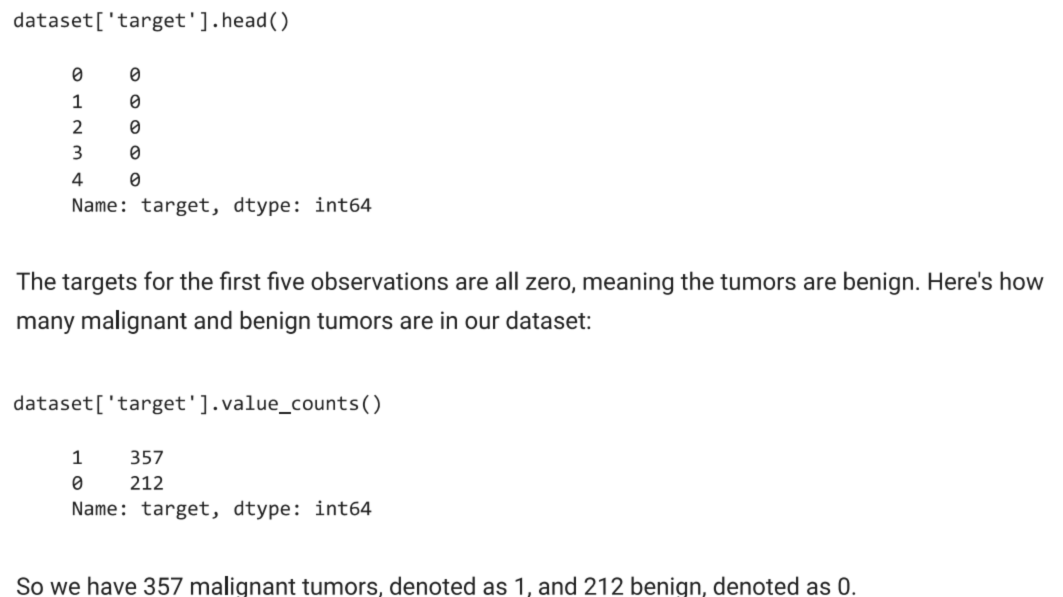
**Aim - Implement Classification, Decision Trees, Random Forest**

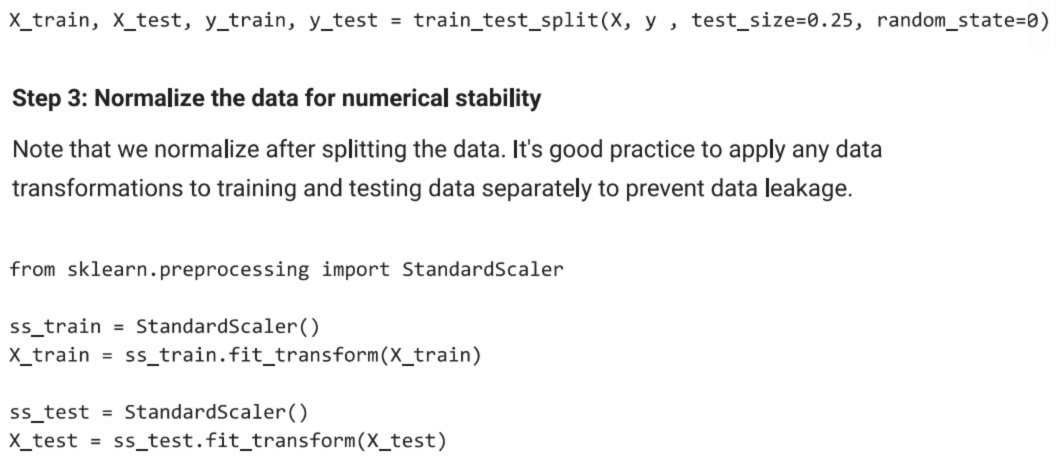
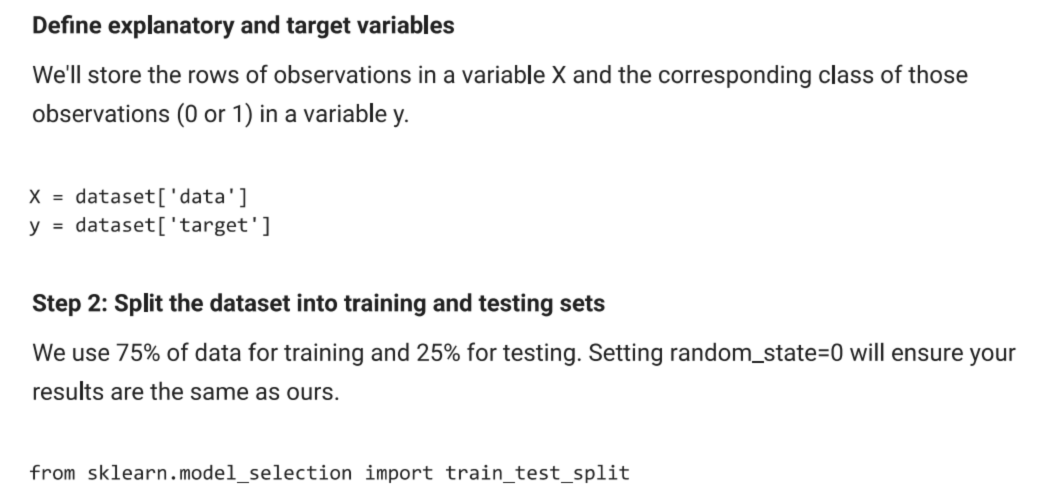


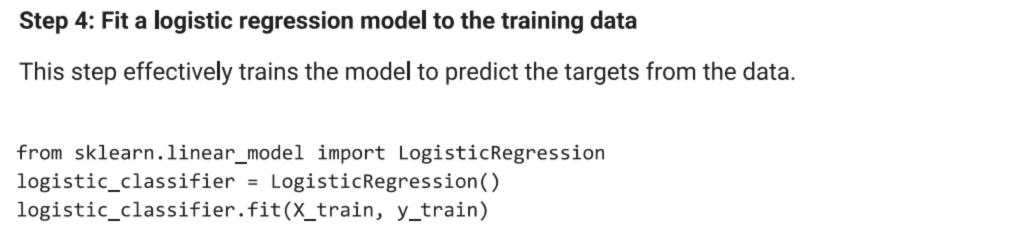


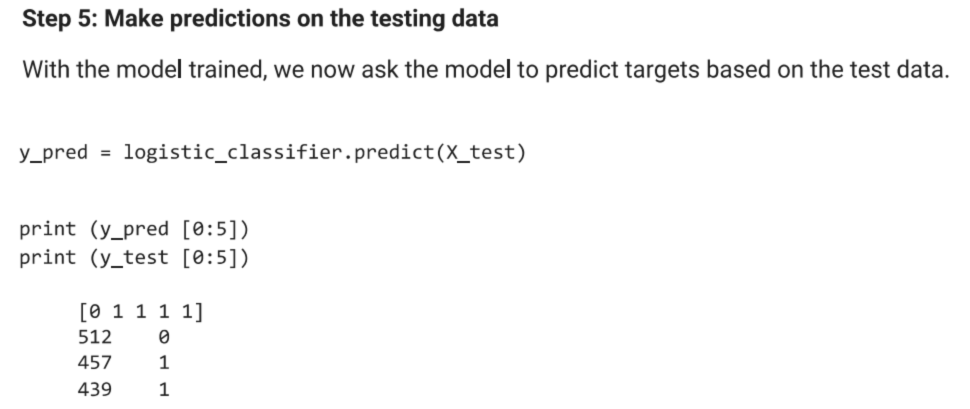


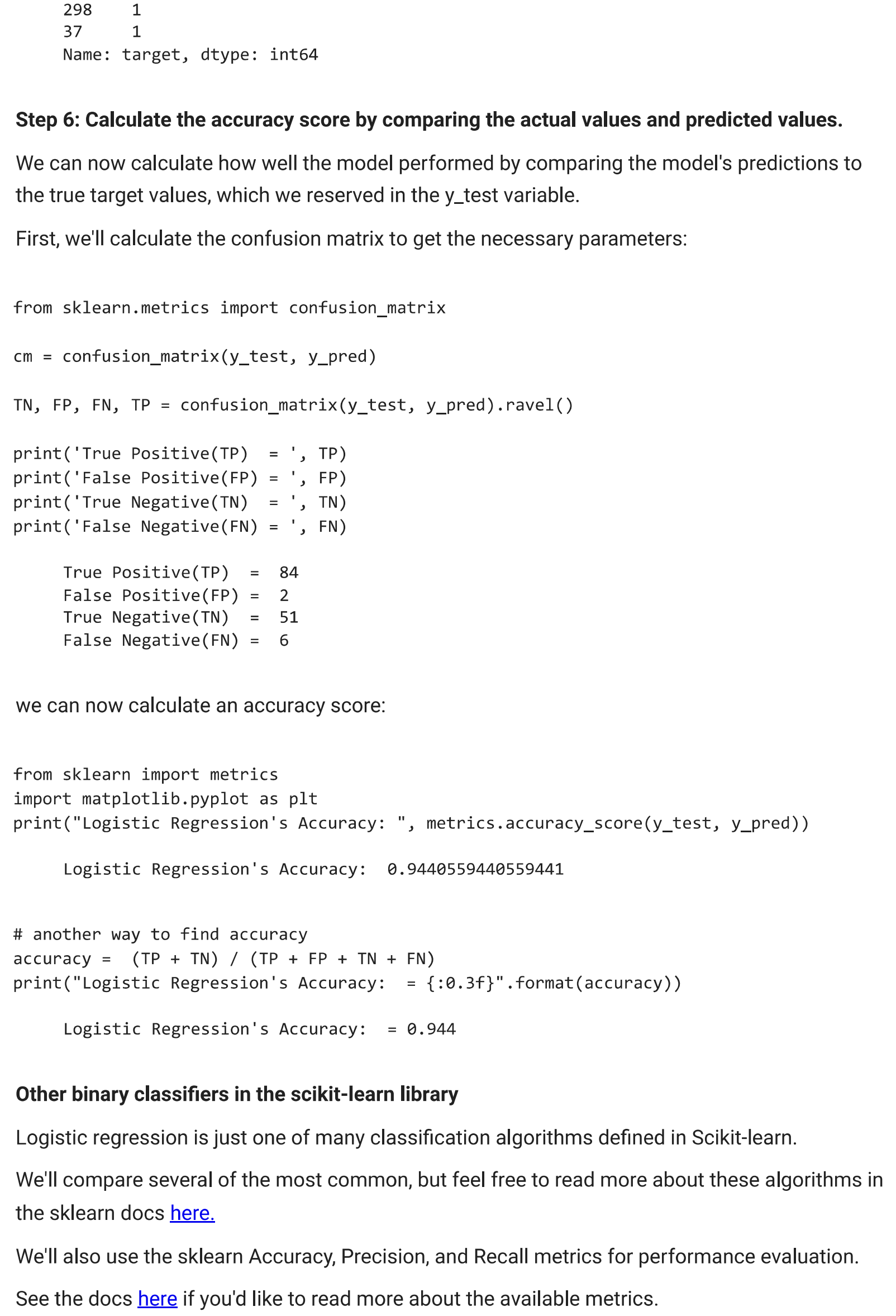


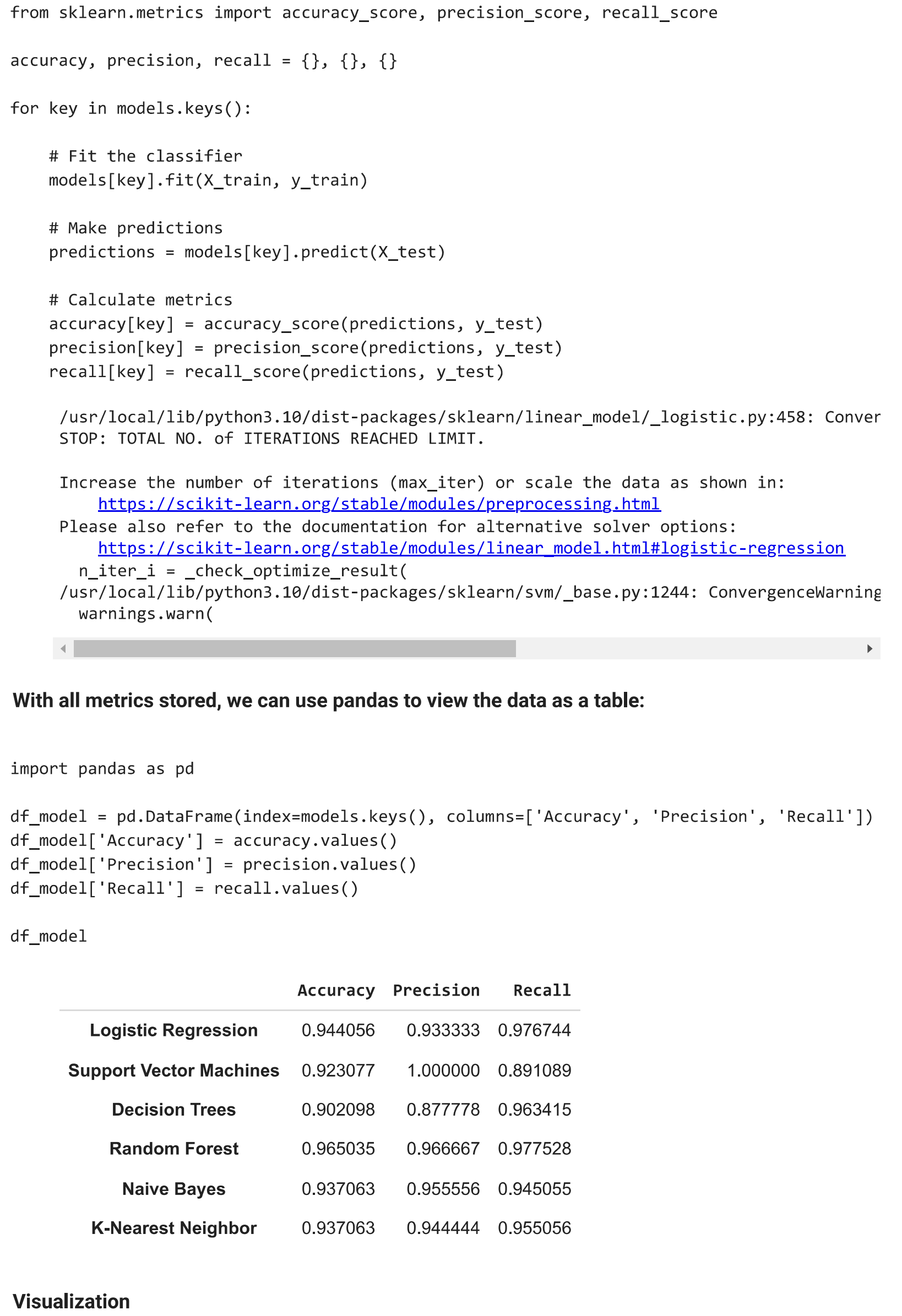
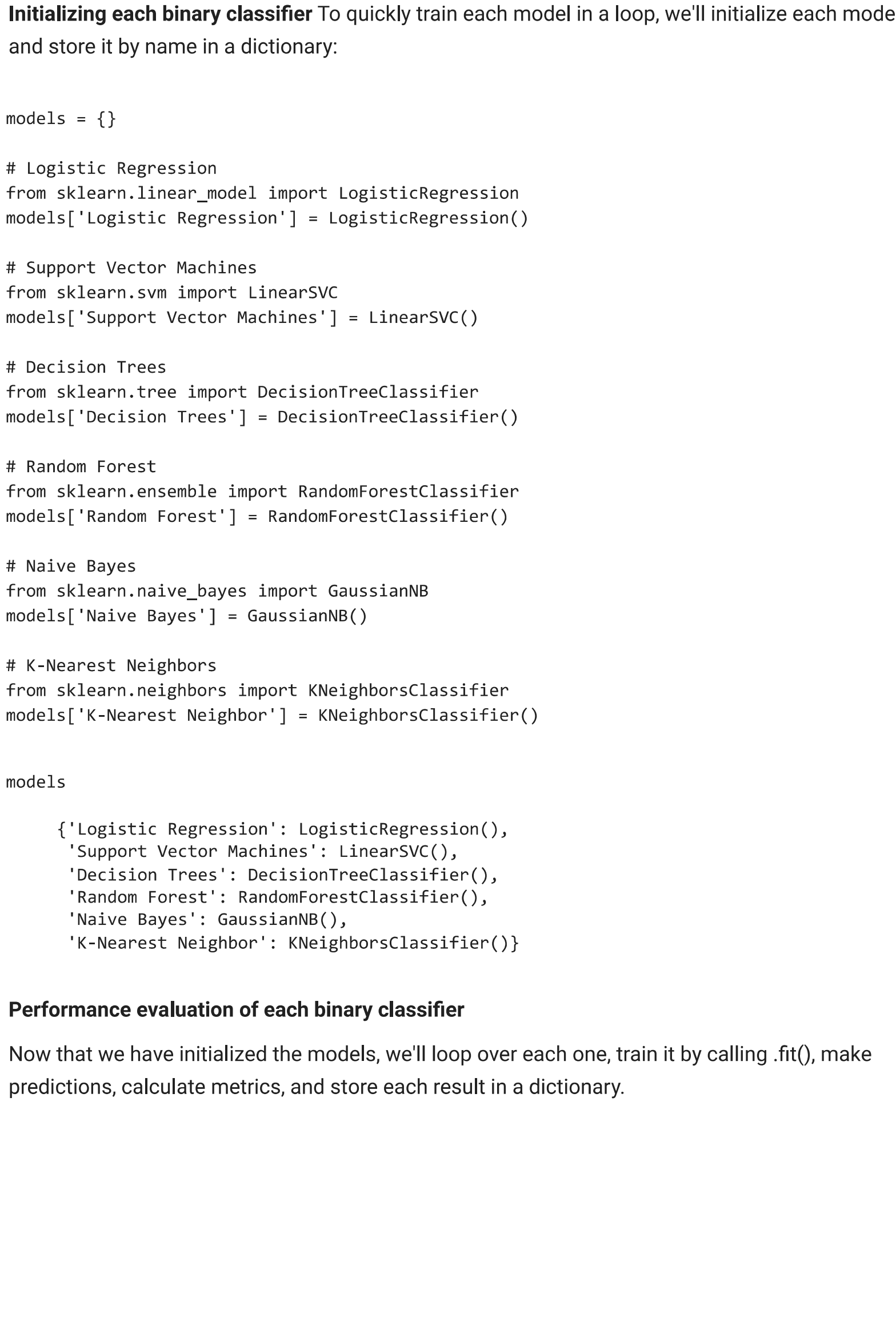


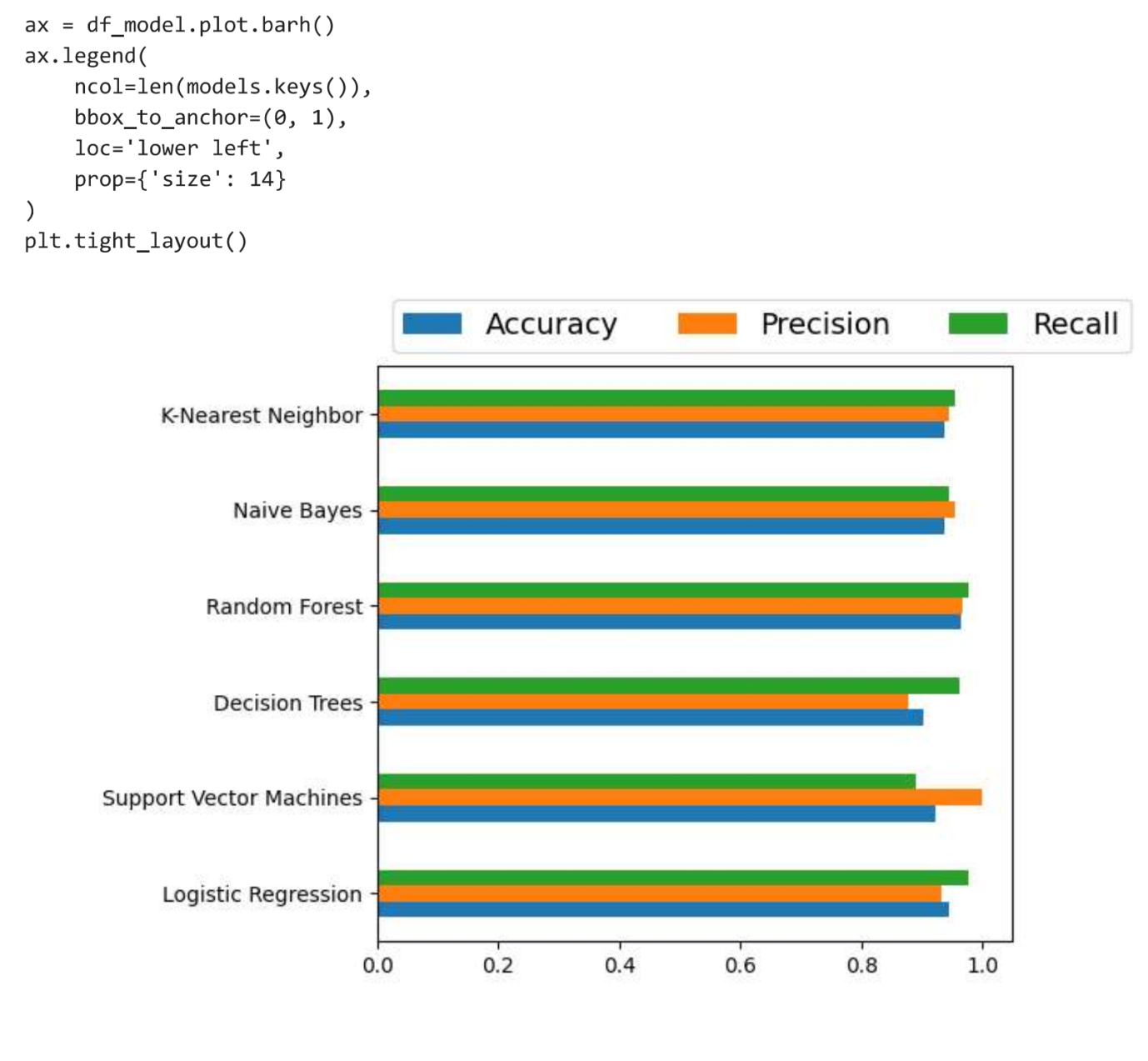












**Practical 9**

**Aim - Implement Clustering**

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

Turn the data into a set of points:

data = list(zip(x, y))

print(data)

Result:

[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10, 21), (12, 21)]

In order to find the best value for K, we need to run K-means across our data for a range of possible values. We only have 10 data points, so the maximum number of clusters is 10. So for each value K in range(1,11), we train a K-means model and plot the intertia at that number of clusters:

inertias = []

for i in range(1,11):

    kmeans = KMeans(n\_clusters=i)

    kmeans.fit(data)

    inertias.append(kmeans.inertia\_)

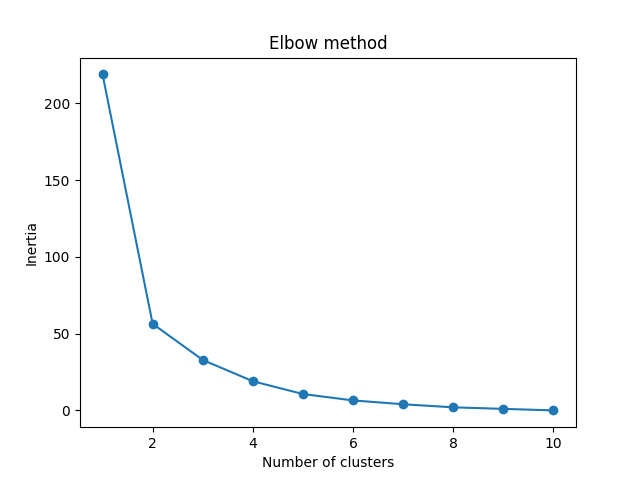
plt.plot(range(1,11), inertias, marker='o')

plt.title('Elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()



We can see that the "elbow" on the graph above (where the interia becomes more linear) is at K=2. We can then fit our K-means algorithm one more time and plot the different clusters assigned to the data:

kmeans = KMeans(n\_clusters=2)

kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels\_)

plt.show()

Result:

