|  |  |
| --- | --- |
| Internship Project Title | Build a Classification Model for Drug Trials Dataset |
| Name of the Company | TCS ION |
| Name of the Industry Mentor | Himdweep Walia |
| Name of the Institute | SRM Institute of Science and Technology |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 12/10/2022 | 09/12/2022 | 225 hours | Python | Google Research Collab , Jupyter Notebook  Library : Pandas , NumPy, Support vector machines (SVMs) , matplotlib.pyplot |

# ACKNOWLEDGEMENTS

I am highly indebted to TCS iON team for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

I would like to express my special gratitude and thanks to my industry mentor **Himdweep Walia** for giving me such attention and time.

I express my humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. I extend my sincere thanks to Director DOE, SRM Institute of Science and Technology, Prof. **Dr Manoranjan Pon Ram,** for his invaluable support. SRM Institute of Science and Technology, for her support throughout the project work. Also I want to convey my thanks to Programme Coordinator **Dr. G. Babu**, Directorate of online Education, SRM Institute of Science and Technology, for their inputs during the project reviews and support. I sincerely thank the Directorate of online Education, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, I would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

**Vaishnavi G**

# ABSTRACT

For a pharmaceutical agency its very important to keep track of their drugs and their effectiveness and to tabulate all drugs that they have sold and account for each drug's effectiveness. This data has been collected over the years and it contains data points such as the drug's name, reviews by customers, popularity , use cases of the drug, and most importantly the side effects contributed by the drug. Numerous medicines that had already been approved for usage had to be taken off the market due to unforeseen side effects.

The objective of this project is to build a classification model that classifies the side effects of a particular drug by age, gender and race which will help to identify potential side effects and minimizing risks during the drug research process.

Drug data which was gathered by academics and compiled in open databases, computational approaches for side effect forecasting will be used to create a model that classifies the trial data of a drug based on their age, gender and race.

1. We will be utilizing a data set consisting of 4,00,000 patients containing the following details for each patient: Name, Age , Gender , Race , Side effects .
2. Clean and Sanitize the dataset .
3. Preprocess the data to perform data partitioning and handle missing values.
4. Create training and testing sets.
5. Build a classifier and fit the data to the model.
6. Check the accuracy of the model and measure its effectiveness.

# SOFTWARE REQUIREMENTS & SPECIFICATION

**System configurations**

The software requirement specification can produce at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by established a complete information description, a detailed functional description, a representation of system behavior, and indication of performance and design constrain, appropriate validate criteria, and other information pertinent to requirements.

**Software Requirements:**

* Operating system : Windows 10
* Coding Language : Python
* Google Research Collab , Jupyter Notebook
* Library : Pandas , NumPy, Support vector machines (SVMs) , matplotlib.pyplot

**Hardware Requirement:**

* System : Intel I5 Processor
* Hard Disk : 1TB.
* RAM : 8GB.

# INTERNSHIP APPROACH

1. We will be utilizing a data set consisting of 4,00,000 patients containing the following details for each patient: Name, Age , Gender , Race , Side effects .
2. Clean and Sanitize the dataset .
3. Preprocess the data to perform data partitioning and handle missing values.
4. Create training and testing sets.
5. V.Build a classifier and fit the data to the model.
6. Check the accuracy of the model and measure its effectiveness.

# INTERNSHIP ACTIVITIES

#### Week 1 : Learned about Data Set Information

The dataset provides patient reviews on specific drugs along with related conditions and a 10 star patient rating reflecting overall patient satisfaction.

1. Sentiment analysis of drug experience over multiple facets, i.e. sentiments learned on specific aspects such as effectiveness and side effects,
2. the transferability of models among domains, i.e. conditions, and
3. the transferability of models among different data sources

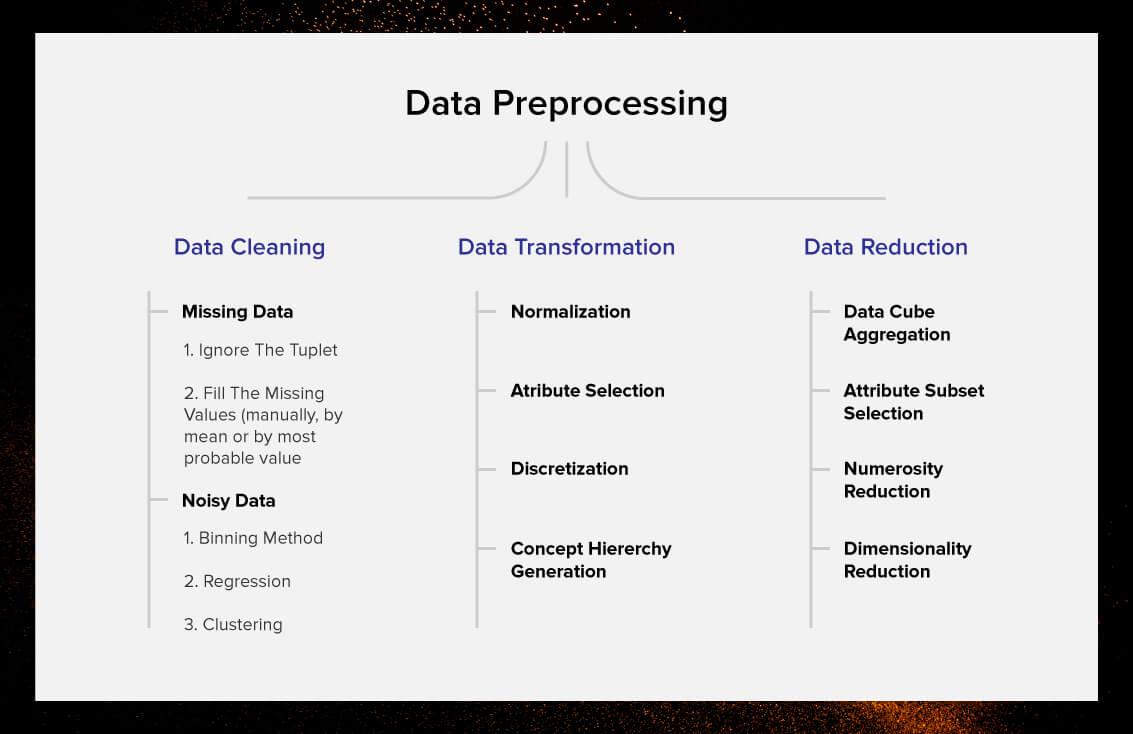
#### **Week 2 : Learned about Data Set Information**

1. What is Data Preprocessing?
2. Why is Data Preprocessing important?
3. Steps in Data Preprocessing
4. Data Integration
5. Data Cleaning
6. Data Transformation
7. Data Reduction or Dimension Reduction
8. Data pre-processing of structured data in machine learning
9. What are Data and Attributes?
10. Machine Learning for Data Analytics

#### Week 3 : 7 Types of Classification Algorithms in detail

1. Stochastic Gradient Descent
2. K-Nearest Neighbours
3. Decision Tree
4. Random Forest

#### Week 4 : Data PreProcessing



**Data pre processing with Python - Commands**

Different data set types with python

1. Missing values
2. Outliers
3. Overfitting
4. Data with no numerical values
5. Different date formats

**Missing values**

* Missing values are a common problem while dealing with data! The values can be missed because of various reasons such as human errors, mechanical errors, etc.
* There are three techniques to solve the missing values’ problem in order to find out the most accurate features, and they are:

1. Dropping
2. Numerical imputation
3. Categorical imputation

* Dropping is the most common method to take care of the missed values. Those rows in the data set or the entire columns with missed values are dropped in order to avoid errors to occur in data analysis.

|  |
| --- |
| #Dropping columns in the data higher than 60% threshold  data = data[data.columns[data.isnull().mean() < threshold]]  #Dropping rows in the data higher than 60% threshold  data = data.loc[data.isnull().mean(axis=1) < threshold] |

* Numerical imputation - The word imputation implies replacing the missing values with such a value that makes sense. And, numerical imputation is done in the data with numbers.

|  |
| --- |
| #For filling all the missed values as 0  data = data.fillna(0)  #For replacing missed values with median of columns  data = data.fillna(data.median()) |

* Categorical imputation - This technique of imputation is nothing but replacing the missed values in the data with the one which occurs the maximum number of times in the column. But, in case there is no such value that occurs frequently or dominates the other values, then it is best to fill the same as “NAN”.

|  |
| --- |
| #Categorical imputation  data['column\_name'].fillna(data['column\_name'].value\_counts().idxmax(), inplace=True) |

**Outliers -**

* An outlier differs significantly from other values and is too distanced from the mean of the values. Such values that are considered outliers are usually due to some systematic errors or flaws.

|  |
| --- |
| #For identifying the outliers with the standard deviation method  outliers = [x for x in data if x < lower or x > upper]  print('Identified outliers: %d' % len(outliers))  #Remove outliers  outliers\_removed = [x for x in data if x >= lower and x <= upper]  print('Non-outlier observations: %d' % len(outliers\_removed)) |

In the codes above, “lower” and “upper” signify the upper and lower limit in the data set.

**Overfitting -**

* Overfitting occurs when the model fits the data too well or simply put when the model is too complex. Overfitting model learns the detail and noise in the training data to such an extent that it negatively impacts the performance of the model on new data/test data.

|  |
| --- |
| data['bin'] = pd.cut(data['value'], bins=[100,250,400,500], labels=["Lowest", "Mid", "High"]) |

#### Week 5 : Data Cleaning

Learned about data Cleaning With Pandas and NumPy ( Commands and Usage).

**What is data cleaning ?**

* When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct.
* Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset.



**Data cleaning cycle**

* It is the method of analyzing, distinguishing, and correcting untidy, raw data. Data cleaning involves filling in missing values, distinguish and fix errors present in the dataset. Whereas the techniques used for data cleaning might vary in step with different types of datasets, the following are standard steps to map out data cleaning:

**Data cleaning with Pandas**

* Pandas stand for “Python Data Analysis Library” .
* Pandas is the popular Python library that is mainly used for data processing purposes like cleaning, manipulation, and analysis.
* It consists of classes to read, process, and write CSV data files.
* There are numerous Data cleaning tools present but, the Pandas library provides a really fast and efficient way to manage and explore data.
* It does that by providing us with Series and DataFrames, which help us not only to represent data efficiently but also manipulate it in various ways.

|  |
| --- |
| #importing module  import pandas as pd |

* To import the dataset we use the **read\_csv()** function of pandas and store it in the DataFrame named as data.
* As the dataset is in tabular format, when working with tabular data in Pandas it will be automatically converted in a **DataFrame**.
* DataFrame is a two-dimensional, mutable data structure in Python. It is a combination of rows and columns like an excel sheet.

|  |
| --- |
| #importing the dataset by reading the csv file  data = pd.read\_csv('Iris.csv')  #displaying the first five rows of dataset  print(data.head()) |

* The head() function is a built-in function in pandas for the dataframe used to display the rows of the dataset.
* We can specify the number of rows by giving the number within the parenthesis. By default, it displays the first five rows of the dataset.
* If we want to see the last five rows of the dataset we use the tail()function of the dataframe like this:

|  |
| --- |
| #displayinf last five rows of dataset  data.tail() |

**Rebuild Missing Data**

To find and fill the missing data in the dataset we will use another function. There are 4 ways to find the null values if present in the dataset.

1. Using isnull() function:

This function provides the boolean value for the complete dataset to know if any null value is present or not.

|  |
| --- |
| data.isnull() |

1. Using isna() function:

|  |
| --- |
| data.isna() |

This is the same as the isnull() function. Ans provides the same output.

1. Using isna().any()

|  |
| --- |
| data.isna().any() |

This function also gives a boolean value if any null value is present or not, but it gives results column-wise, not in tabular format.

1. Using isna(). sum()

|  |
| --- |
| data.isna().sum() |

This function gives the sum of the null values preset in the dataset column-wise.

1. Using isna().any().sum()

|  |
| --- |
| data.isna().any().sum() |

This function gives output in a single value if any null is present or not.

If there are any null value s preset we can fill those places with any other value using the fillna() function of DataFrame.Following is the syntax of fillna() function:

|  |
| --- |
| DataFrame\_name.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None) |

This function will fill NA/NaN or 0 values in place of null spaces.

**Standardization and Normalization**

* Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
* Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

**De-Duplicate**

* De-Duplicate means remove all duplicate values. There is no need for duplicate values in data analysis. These values only affect the accuracy and efficiency of the analysis result.
* To find duplicate values in the dataset we will use a simple dataframe function i.e. duplicated().

|  |
| --- |
| data.duplicated() |

**Export Dataset**

This is the last step of the data cleaning process. After performing all the above operations, the data is transformed into clean the dataset and it is ready to export for the next process in Data Science or Data Analysis.

#### Week 6 : Data Transformation

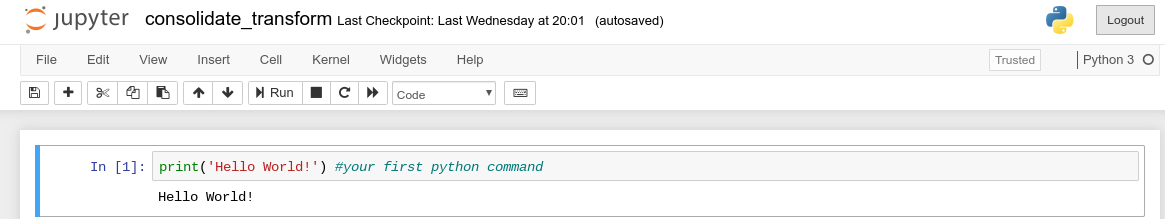
Learned about data transformation process using a live Python use case scenario

Five steps that are mandatory in any research with data:

1. Set your analytics goal – what is a real-life problem that needs research
2. Collect data – find sources of data
3. Prepare data – check data quality, integrity, completeness and perform data cleaning (avoid GIGO – “Garbage In, Garbage Out” situation)
4. Analyze data – the fun part
5. Interpret results – understood results, do a reality check, go back to phase 2 or 3 if needed, do data visualization, prepare the next actions, etc.

**Data transformation using Python:**

Step 1 : Create Your Python Script



Step 2 : Import a Pandas Module

To load the CSV files into your notebook, you need to import a Pandas module, which is an add-on file that extends Python’s basic functionalities. To do this, simply type “import pandas as pd”.

IMG_256

Step 3: Load file content

pd.read\_csv( ‘/home/jacekpolewski/tutorial\_files/data\_file\_1.csv’, sep=’;’)

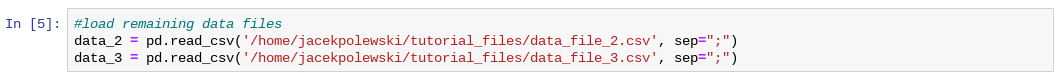
Step 4: Assign Data Content to a Variable



Step 5: Inspect Data Using the Head() Function

After loading your data and assigning it to a variable, it’s a good idea to inspect the data to see if all of the content has loaded correctly. You can do this by looking at the first few rows of the data using the Python head() function.

Step 6: Consolidate multiple Datasets



Step 7 : Data Transformation/Preparation

If you need to modify your data or transform it into your desired format before extracting valuable insights, Python and Pandas make it as fast and easy as possible to do so.

If you need to create a new column, just type in and execute data\_comb[‘Test’] = ‘hello’. When you now inspect the data, you will notice a new column called ‘Test’ filled with values ‘hello’.



#### **Week 7 :Create training and testing dataset in python**

**What is train test split ?**

* The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications.
* This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results.
* By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data.

**Prerequisites for Train and Test Data**

* We will need the following Python libraries for this tutorial- pandas and sklearn.We can install these with pip- commands

|  |
| --- |
| pip install pandas  pip install sklearn |

* We use pandas to import the dataset and sklearn to perform the splitting. You can import these packages as-

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.datasets import load\_iris |

**How to Split Train and Test Set in Python Machine Learning?**

1. Loading the Dataset

Let’s load the csv dataset using pandas.

|  |
| --- |
| data=pd.read\_csv('forestfires.csv')  data.head() |

1. Splitting

Let’s split this data into labels and features. Now, what’s that? Using features, we predict labels. I mean using features (the data we use to predict labels), we predict labels (the data we want to predict).

|  |
| --- |
| y=data.temp  x=data.drop('temp',axis=1) |

Temp is a label to predict temperatures in y; we use the drop() function to take all other data in x. Then, we split the data.

|  |
| --- |
| x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)  x\_train.head() |

1. Plotting of Train and Test Set in Python

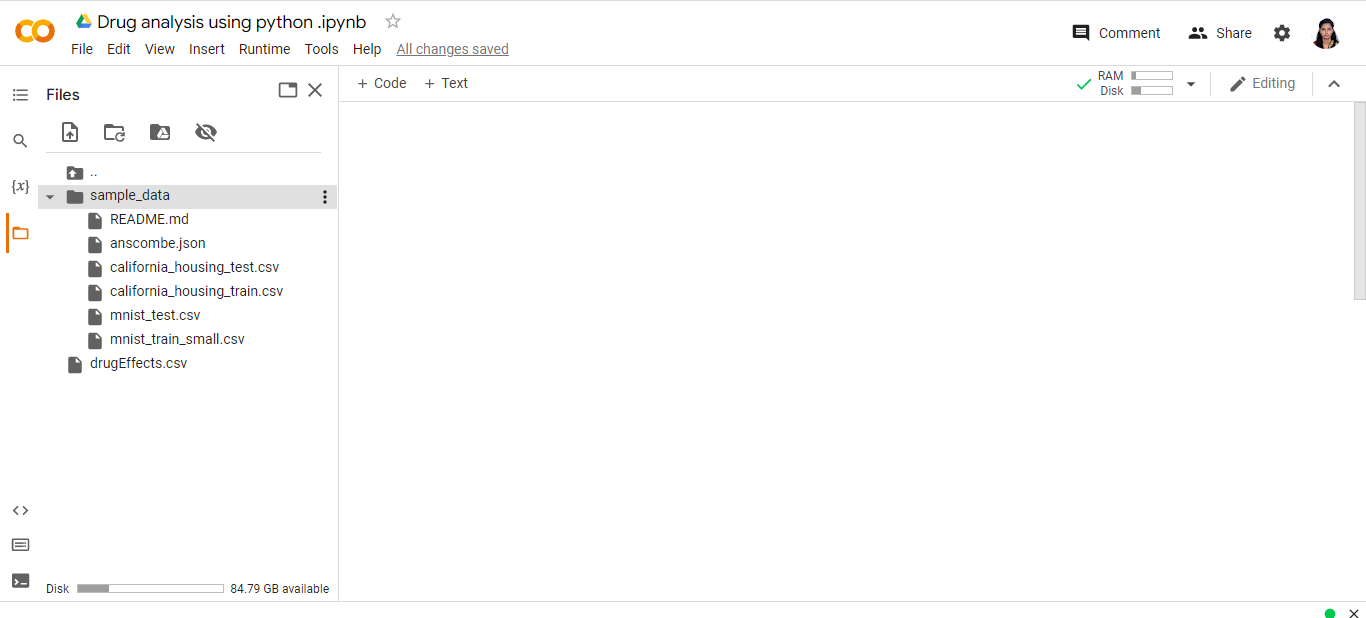
We fit our model on the train data to make predictions on it. Let’s import the linear\_model from sklearn, apply linear regression to the dataset, and plot the results.

|  |
| --- |
| from sklearn.linear\_model import LinearRegression as lm  model=lm().fit(x\_train,y\_train)  predictions=model.predict(x\_test)  import matplotlib.pyplot as plt  plt.scatter(y\_test,predictions) |

**Internship Handson**

**Import related modules and dataset file**

Upload the dataset in colab.



**Import required libraries in python:**

|  |
| --- |
| **import pandas as pd** |

pandas (all lowercase) is a popular Python-based data analysis toolkit which can be imported using import pandas as pd.

It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array.

This makes pandas a trusted ally in data science and machine learning.

**Import** = “Bring this functionality or library to my python script”

**Pandas** = The library you want to import, in this case, it’s pandas

**As** = The python nomenclature for creating as alias. This is a fancy way of taking a long word and referencing it as a short word

**pd** = The standard short name for referencing pandas

|  |
| --- |
| **import numpy as np** |

NumPy, which stands for Numerical Python, is a scientific computing library built on top of the Python programming language.

The import numpy portion of the code tells Python to bring the NumPy library into your current environment.

The as np portion of the code then tells Python to give NumPy the alias of np. This allows you to use NumPy functions by simply typing np.function\_name rather than numpy.function\_name.

|  |
| --- |
| **from sklearn.svm import SVC** |

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

It is C-support vector classification whose implementation is based on libsvm. The module used by scikit-learn is sklearn.svm.SVC. This class handles the multiclass support according to one-vs-one scheme.

|  |
| --- |
| **from sklearn.ensemble import RandomForestClassifier** |

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

|  |
| --- |
| **from sklearn.model\_selection import train\_test\_split** |

The train\_test\_split function of the sklearn.model\_selection package in Python splits arrays or matrices into random subsets for train and test data, respectively.

To use the train\_test\_split function, we’ll import it into our program

|  |
| --- |
| **from sklearn.preprocessing import StandardScaler** |

The StandardScaler function of sklearn is based on the theory that the dataset's variables whose values lie in different ranges do not have an equal contribution to the model's fit parameters and training function and may even lead to bias in the predictions made with that model.

Therefore, before including the features in the machine learning model.

|  |
| --- |
| **from sklearn.metrics import classification\_report** |

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.

|  |
| --- |
| **import matplotlib.pyplot as plt** |

matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB.

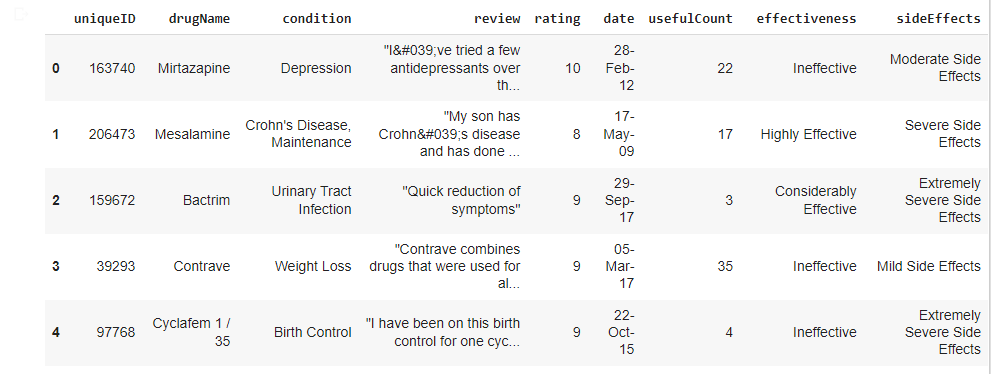
Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes

**Code:**



**Output:**



**List data and check ratings of drug**

|  |
| --- |
| **df = pd.read\_csv('drugEffects.csv')**  **df.head()** |

**df -** A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**read\_csv()** - Loading CSV files specifically with pandas

**head() -** The head function in Python displays the first five rows of the dataframe by default. It takes in a single parameter. the number of rows. We can use this parameter to display the number of rows of our choice.

**Pandas tolist() :**

Pandas tolist() is used to convert a series to list. Initially the series is of type pandas.core.series.Series and applying tolist() method, it is converted to list data type.

**Syntax:**

|  |
| --- |
| **Series.tolist()**  **Return type: Converted series into List** |

**Code:**

|  |
| --- |
|  |

**Output :**



**Check effectiveness of drug and value counts**

**To check the effectiveness of the drugs**

|  |
| --- |
| df['effectiveness'].value\_counts().plot(kind='pie')  plt.show() |

**df -** A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Value counts() -**

pandas is an open-source Python library that provides operations to analyze and manipulate data structures called data frames. The value\_counts() function in pandas returns a series that contains the number of unique values. A series is a one-dimensional array.

**Parameters**

**The value\_counts function takes the following parameters:**

* **normalize (optional):** If set to True, the function returns the relative frequencies of the values. It is set to False by default.
* **sort (optional):** If set to True, the function returns the values in a sorted manner. It is set to True by default.
* **ascending (optional):** If set to True, values are sorted in an ascending manner. It is set to False by default.
* **bins (optional):** Groups values into bins instead of counting them. It is set to None by default.
* **dropna (optional):** If set to True, counts of NaN are not included. It is set to True by default.

**Return value:**

The function returns a series of counts of unique values.

**plot() :**

Pandas uses the plot() method to create diagrams. We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

**Scatter:**

* a scatter plot with the kind argument:
* kind = 'scatter'
* A scatter plot needs an x- and a y-axis.
* Include the x and y arguments like this: x = 'Duration', y = 'Calories'

**Histogram :**

* Use the kind argument to specify that you want a histogram:
* kind = 'hist'
* A histogram needs only one column.
* A histogram shows us the frequency of each interval

**To check the effectiveness of the drugs**

|  |
| --- |
| df['effectiveness'].value\_counts().plot(kind='pie')  plt.show() |

**df -** A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Value counts() -**

pandas is an open-source Python library that provides operations to analyze and manipulate data structures called data frames. The value\_counts() function in pandas returns a series that contains the number of unique values. A series is a one-dimensional array.

**The value\_counts function takes the following parameters:**

* **normalize (optional):** If set to True, the function returns the relative frequencies of the values. It is set to False by default.
* **sort (optional):** If set to True, the function returns the values in a sorted manner. It is set to True by default.
* **ascending (optional):** If set to True, values are sorted in an ascending manner. It is set to False by default.
* **bins (optional):** Groups values into bins instead of counting them. It is set to None by default.
* **dropna (optional):** If set to True, counts of NaN are not included. It is set to True by default.

**Return value:**

The function returns a series of counts of unique values.

**plot() :**

Pandas uses the plot() method to create diagrams. We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

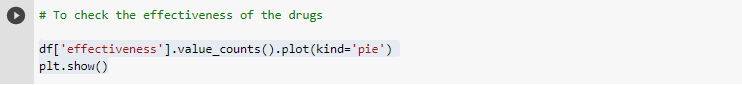
**Scatter:**

* a scatter plot with the kind argument:
* kind = 'scatter'
* A scatter plot needs an x- and a y-axis.
* Include the x and y arguments like this: x = 'Duration', y = 'Calories'

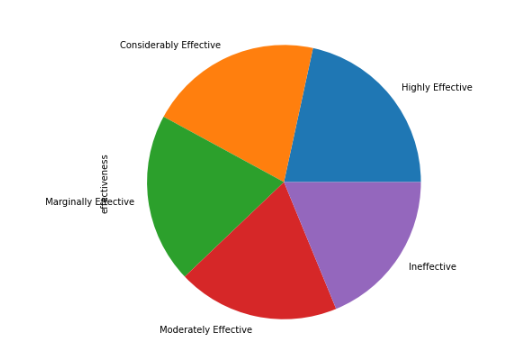
**Histogram :**

* Use the kind argument to specify that you want a histogram:
* kind = 'hist'
* A histogram needs only one column.
* A histogram shows us the frequency of each interval

**Code :**



**Output:**



**Various conditions the medicine cures (top 10)**

To check the various conditions the medicine cures (top 10) since they are over thousand

|  |
| --- |
| **df['condition'].value\_counts().head(10).plot(kind='bar')**  **plt.xlabel('condition')**  **plt.show()** |

**df** - A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Value counts() -** pandas is an open-source Python library that provides operations to analyze and manipulate data structures called data frames. The value\_counts() function in pandas returns a series that contains the number of unique values.

**plot() :**

Pandas uses the plot() method to create diagrams. We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

**kind='bar'**

The plot.bar() function is used to vertical bar plot.

A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the values that they represent.

**plt.xlabel('')**

The xlabel() function in pyplot module of matplotlib library is used to set the label for the x-axis..

**Pyplot**

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

|  |
| --- |
| **import matplotlib.pyplot as plt** |

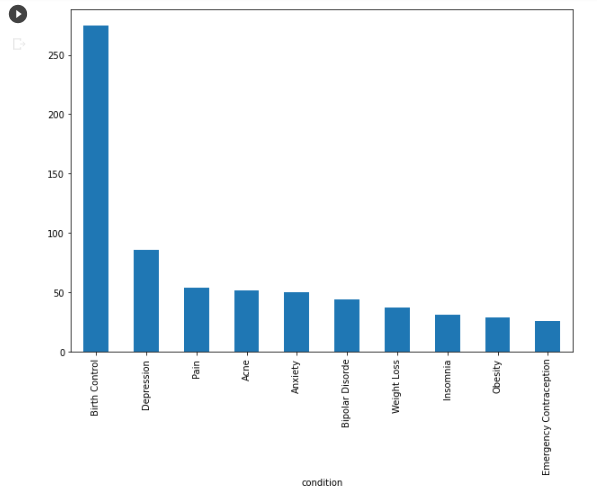
**plt.show()**

The show() function in pyplot module of matplotlib library is used to display all figures.

**Code :**

|  |
| --- |
|  |

**Output:**



**Check the conditions cured by the drugs:**

|  |
| --- |
| df['condition'].value\_counts().head(10).plot(kind='bar')  plt.('condition')  plt.ylabel('drugName')  plt.show() |

**df** - A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Value counts() -** pandas is an open-source Python library that provides operations to analyze and manipulate data structures called data frames. The value\_counts() function in pandas returns a series that contains the number of unique values. A series is a one-dimensional array.

**(kind='bar')**

The plot.bar() function is used to vertical bar plot.

A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the values that they represent. A bar plot shows comparisons among discrete categories. One axis of the plot shows the specific categories being compared, and the other axis represents a measured value.

**plot() :**

Pandas uses the plot() method to create diagrams. We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

**plt.show()**

The show() function in pyplot module of matplotlib library is used to display all figures.

**xlabel('')**

The xlabel() function in pyplot module of matplotlib library is used to set the label for the x-axis..

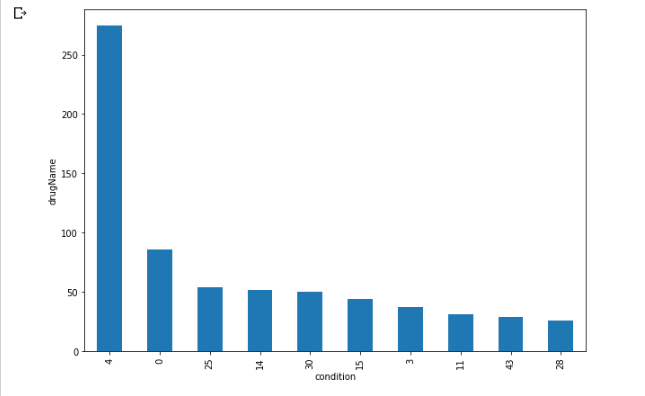
**ylabel('')**

The ylabel() function in pyplot module of matplotlib library is used to set the label for the x-axis..

**Code :**

|  |
| --- |
|  |

**Output:**



|  |
| --- |
| **Link to code and executable file**  Link to run the code directly in browser: <https://colab.research.google.com/drive/1gFZVnUoIvErA5dkvEAZMzsRjUHezFPv_?usp=sharing>  Project codes : |