Harshit Mishra(19BCE0799)

CSE4020 – MACHINE LEARNING

LAB ASSIGNMENT 1

Ques: Demonstrate possible missing value analysis approaches using any real-world data.

Dataset Used: Train dataset containing Row ID, Order ID, Order Quantity, Sales and Profit attributes.

Procedure:

- We first import the dataset into our workspace using pandas.
- We then find the attributes having null values in them.
- Next, we print first few rows of the dataset to know what is the value used for missing data, i.e., Nan/Null/Blank/0 or -1.
- Then we check how many null values are there in each attribute.
- Now we replace all the missing values either with mean of non-null values or by median of non-null values.
- We then see which filling type has better consistency with data set, that is, mean or median.
- Finally, we replace the missing value with the better fills…

Harshit Mishra(19BCE0799)

Code:

This is to fill the null values with mean of non-null values

```
# Importing the Libraries
import pandas as pd

# Importing the dataset
train = pd.read_csv("train.csv")

# Printing the dataset
train

# Getting the info of dataset
train.info()

# Printing first few rows of dataset
train.head(10)

# Checking the rows containing null values
train.isnull().sum()

# Filling the missing values of Order Quantity Attribute
mean_value = train['Order Quantity'].mean()
train['Order Quantity'] = train['Order Quantity'].fillna(mean_value)

# Again checking if there are rows with null values
train.isnull().sum()

# Filling the missing values of Sales Attribute
mean_value = train['Sales'].mean()
train['Sales'] = train['Sales'].fillna(mean_value)

# Again checking if there are rows with null values
train.isnull.sum()

# Filling the missing values of profit attribute
mean_value = train['profit'].mean()
train['profit'] = train['profit'].fillna(mean_value)

# Again checking if there still are missing values
train.isnull.sum()
```

Harshit Mishra(19BCE0799)

This is to fill the null values with median of non-null values

```
# Importing the Libraries
import pandas as pd

# Importing the dataset
train = pd.read_csv("train.csv")

# Printing the dataset
train

# Getting the info of dataset
train.info()

# Printing first few rows of dataset
train.head(10)

# Checking the rows containing null values
train.isnull().sum()

# Filling the missing values of Order Quantity Attribute
median_value = train['Order Quantity'].median()
train['Order Quantity'] = train['Order Quantity'].fillna(mean_value)

# Again checking if there are rows with null values
train.isnull().sum()

# Filling the missing values of Sales Attribute
median_value = train['Sales'].median()
train['Sales'] = train['Sales'].fillna(median_value)

# Again checking if there are rows with null values
train.isnull.sum()

# Filling the missing values of profit attribute
median_value = train['profit'].mean()
train['profit'] = train['profit'].fillna(median_value)

# Again checking if there still are missing values
train.isnull.sum()
```

Harshit Mishra(19BCE0799)

Code Snippet and Explanation:

Filling with mean value:

```
In [1]: # Importing the Libraries
        import pandas as pd

In [2]: # Importing the dataset
        train = pd.read_csv("train.csv")

In [3]: # Getting the info of dataset
        train.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 50 entries, 0 to 49
        Data columns (total 5 columns):
         #   Column          Non-Null Count  Dtype
        ---  ------          --------------  -----
         0   Row ID          50 non-null     int64
         1   Order ID        50 non-null     int64
         2   Order Quantity  42 non-null     float64
         3   Sales           41 non-null     float64
         4   profit          45 non-null     float64
        dtypes: float64(3), int64(2)
        memory usage: 2.1 KB
```

Since there are 50 data rows but the non-null rows in Order Quantity, Sales and profit are less than 50, we can conclude that there are null values in these attributes.

```
In [4]: # Printing first few rows of dataset
        train.head(10)

Out[4]:
```

	Row ID	Order ID	Order Quantity	Sales	profit
0	1	3	7.0	261.5400	0.80
1	2	6	6.0	-6.9300	NaN
2	3	32	-90.0	2808.0800	0.65
3	4	32	NaN	1761.4000	0.72
4	5	32	NaN	NaN	0.60
5	6	32	15.0	140.5600	0.60
6	7	35	-30.0	288.5600	NaN
7	8	35	14.0	1892.8480	NaN
8	9	36	46.0	2484.7455	0.55
9	10	65	-32.0	NaN	0.49

From here we can know the missing values are assigned with NaN value.

```
In [5]: # Checking the rows containing null values
        train.isnull().sum()

Out[5]: Row ID          0
        Order ID        0
        Order Quantity  8
        Sales           9
        profit          5
        dtype: int64
```

This cell informs us that there are 8 null values in Order Quantity, 9 in Sales and 5 in profit.

Harshit Mishra(19BCE0799)

```
In [6]: # Filling the missing values of Order Quantity Attribute
        mean_value = train['Order Quantity'].mean()
        train['Order Quantity']=train['Order Quantity'].fillna(mean_value)

In [7]: # Again checking if there are rows with null values
        train.isnull().sum()

Out[7]: Row ID            0
        Order ID          0
        Order Quantity    0
        Sales             9
        profit            5
        dtype: int64

In [8]: # Filling the missing values of Sales Attribute
        mean_value = train['Sales'].mean()
        train['Sales'] = train['Sales'].fillna(mean_value)

In [9]: # Again checking if there are rows with null values
        train.isnull().sum()

Out[9]: Row ID            0
        Order ID          0
        Order Quantity    0
        Sales             0
        profit            5
        dtype: int64
```

```
In [10]: # Filling the missing values of profit Attribute
         mean_value = train['profit'].mean()
         train['profit'] = train['profit'].fillna(mean_value)

In [11]: # Again checking if there still are missing values
         train.isnull().sum()

Out[11]: Row ID            0
         Order ID          0
         Order Quantity    0
         Sales             0
         profit            0
         dtype: int64
```

Now we have filled all the null values in our dataset…

```
In [12]: # Printing the rows to check what values are filled
         train.head(10)
```

Out[12]:

	Row ID	Order ID	Order Quantity	Sales	profit
0	1	3	7.000000	261.540000	0.800000
1	2	6	6.000000	-6.930000	0.554667
2	3	32	-90.000000	2808.080000	0.650000
3	4	32	19.738095	1761.400000	0.720000
4	5	32	19.738095	1331.872098	0.600000
5	6	32	15.000000	140.560000	0.600000
6	7	35	-30.000000	288.560000	0.554667
7	8	35	14.000000	1892.848000	0.554667
8	9	36	46.000000	2484.745500	0.550000
9	10	65	-32.000000	1331.872098	0.490000

The values with which null values are replaced-

In Order Quantity ---> 19.738095

In Sales ---> 1331.872098

In profit ---> 0.554667

Harshit Mishra(19BCE0799)

Filling with median value:

In [1]:
```
# Importing the Libraries
import pandas as pd
```

In [2]:
```
# Importing the dataset
train = pd.read_csv("train.csv")
```

In [3]:
```
# Getting the info of dataset
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Row ID          50 non-null     int64  
 1   Order ID        50 non-null     int64  
 2   Order Quantity  42 non-null     float64
 3   Sales           41 non-null     float64
 4   profit          45 non-null     float64
dtypes: float64(3), int64(2)
memory usage: 2.1 KB
```

In [4]:
```
# Printing first few rows of dataset
train.head(10)
```

Out[4]:

	Row ID	Order ID	Order Quantity	Sales	profit
0	1	3	7.0	261.5400	0.80
1	2	6	6.0	-6.9300	NaN
2	3	32	-90.0	2808.0800	0.65
3	4	32	NaN	1761.4000	0.72
4	5	32	NaN	NaN	0.60
5	6	32	15.0	140.5600	0.60
6	7	35	-30.0	288.5600	NaN
7	8	35	14.0	1892.8480	NaN
8	9	36	46.0	2484.7455	0.55
9	10	65	-32.0	NaN	0.49

In [5]:
```
# Checking the rows containing null values
train.isnull().sum()
```

Out[5]:
```
Row ID            0
Order ID          0
Order Quantity    8
Sales             9
profit            5
dtype: int64
```

In [6]:
```
# Filling the missing values of Order Quantity Attribute
median_value = train['Order Quantity'].median()
train['Order Quantity']=train['Order Quantity'].fillna(median_value)
```

In [7]:
```
# Again checking if there are rows with null values
train.isnull().sum()
```

Out[7]:
```
Row ID            0
Order ID          0
Order Quantity    0
Sales             9
profit            5
dtype: int64
```

In [8]:
```
# Filling the missing values of Sales Attribute
median_value = train['Sales'].median()
train['Sales'] = train['Sales'].fillna(median_value)
```

Harshit Mishra(19BCE0799)

```
In [9]: # Again checking if there are rows with null values
        train.isnull().sum()

Out[9]: Row ID            0
        Order ID          0
        Order Quantity    0
        Sales             0
        profit            5
        dtype: int64

In [10]: # Filling the missing values of profit Attribute
         median_value = train['profit'].median()
         train['profit'] = train['profit'].fillna(median_value)

In [11]: # Again checking if there still are missing values
         train.isnull().sum()

Out[11]: Row ID            0
         Order ID          0
         Order Quantity    0
         Sales             0
         profit            0
         dtype: int64
```

```
In [12]: # Printing the rows to check what values are filled
         train.head(10)
```

Out[12]:

	Row ID	Order ID	Order Quantity	Sales	profit
0	1	3	7.0	261.5400	0.80
1	2	6	6.0	-6.9300	0.58
2	3	32	-90.0	2808.0800	0.65
3	4	32	24.0	1761.4000	0.72
4	5	32	24.0	329.0300	0.60
5	6	32	15.0	140.5600	0.60
6	7	35	-30.0	288.5600	0.58
7	8	35	14.0	1892.8480	0.58
8	9	36	46.0	2484.7455	0.55
9	10	65	-32.0	329.0300	0.49

The values with which null values are replaced-

In Order Quantity ---> 24

In Sales ---> 329.0300

In profit ---> 0.58

Harshit Mishra(19BCE0799)

Result and Conclusion:

- The result seems to be more consistent when we use median value to fill missing values of Order Quantity and mean value for Sales. The median and mean value for profit attribute is near about same, and thus we can use either of them.

The first 10 rows of train dataset after using median for Order Quantity, mean for Sales and median for profit is-

	Row ID	Order ID	Order Quantity	Sales	profit
0	1	3	7.0	261.540000	0.80
1	2	6	6.0	-6.930000	0.58
2	3	32	-90.0	2808.080000	0.65
3	4	32	24.0	1761.400000	0.72
4	5	32	24.0	1331.872098	0.60
5	6	32	15.0	140.560000	0.60
6	7	35	-30.0	288.560000	0.58
7	8	35	14.0	1892.848000	0.58
8	9	36	46.0	2484.745500	0.55
9	10	65	-32.0	1331.872098	0.49

Harshit Mishra(19BCE0799)

LAB ASSIGNMENT 2

Ques: Build a classifier using decision tree to predict the COVID-19 severity.

Dataset Used: https://www.kaggle.com/hemanthhari/symptoms-and-covid-presence

Procedure:

- We first import the dataset into our workspace using pandas.
- We then specify the attributes to be used as independent attributes and those to be used as dependent attribute.
- Then we encode categorical variables into numbers.
- Then we split our dataset into training set and test set in order to later test the accuracy of our model.
- Then we create an instance of our decision tree classifier.
- We fit our training sets to the object of decision tree classifier in order to train it.
- We then predict the result of our test set and store it in another array.
- Finally, we create the confusion matrix and check the performance of our classifier.

Harshit Mishra(19BCE0799)

Code:

```
# Importing the libraries
import numpy as np
import pandas as pd

# Importing the dataset
data = pd.read_csv("CovidDataset.csv")
X = data.iloc[:, 0:6].values
y = data.iloc[:, 20:].values

# Encoding the Categorical Attribute
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
X[:, 0] = LabelEncoder().fit_transform(X[:, 0])
X[:, 1] = LabelEncoder().fit_transform(X[:, 1])
X[:, 2] = LabelEncoder().fit_transform(X[:, 2])
X[:, 3] = LabelEncoder().fit_transform(X[:, 3])
X[:, 4] = LabelEncoder().fit_transform(X[:, 4])
X[:, 5] = LabelEncoder().fit_transform(X[:, 5])
y[:, 0] = LabelEncoder().fit_transform(y[:, 0])
y=y.astype('int')

# Splitting the dataset into the training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Fitting Decision Tree Classification model to the training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=5)
classifier.fit(X_train, y_train)

# Predicting result for test set
y_pred = classifier.predict(X_test)

# Printing the accuracy of our model
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)

# Printing the classification error
1-accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)

# Printing the Sensitivity
from sklearn.metrics import recall_score
recall_score(y_test, y_pred)
```

```
# Printing the Precision
from sklearn.metrics import precision_score
precision_score(y_test, y_pred)

# Making confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Code Snippet and Explanation:

```
In [1]: # Importing the libraries
        import numpy as np
        import pandas as pd

In [2]: # Importing the dataset
        data = pd.read_csv("CovidDataset.csv")
        X = data.iloc[:, 0:6].values
        y = data.iloc[:, 20:].values
```

Instead of using all the attributes to train our model, we have used only six of them. They are Breathing Problem, Fever, Dry Cough, Sore Throat, Running Nose and Asthma.

The last column is our label attribute and it tells if a person with given symptoms had COVID-19 or not.

```
In [3]: # Encoding the Categorical Attribute
        from sklearn.preprocessing import OneHotEncoder, LabelEncoder
        X[:, 0] = LabelEncoder().fit_transform(X[:, 0])
        X[:, 1] = LabelEncoder().fit_transform(X[:, 1])
        X[:, 2] = LabelEncoder().fit_transform(X[:, 2])
        X[:, 3] = LabelEncoder().fit_transform(X[:, 3])
        X[:, 4] = LabelEncoder().fit_transform(X[:, 4])
        X[:, 5] = LabelEncoder().fit_transform(X[:, 5])
        y[:, 0] = LabelEncoder().fit_transform(y[:, 0])
        y=y.astype('int')
```

Since the attributes contain categorical values of Yes and No, we needed to encode them into numbers for our machine learning model to understand them and thus we have encoded all the categorical attributes into numbers.

```
In [4]: # Splitting the dataset into the training set and test set
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Here we have split our dataset into training set and test set. The training set will be used to train our decision tree classifier and the test will help us analyse the efficacy of trained model.

```
In [5]: # Fitting Decision Tree Classification model to the training set
        from sklearn.tree import DecisionTreeClassifier
        classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=5)
        classifier.fit(X_train, y_train)

Out[5]: DecisionTreeClassifier(criterion='entropy', random_state=5)
```

Here we have trained our decision tree classifier with training dataset. Also, we have set the criterion to entropy and thus the decision tree that we constructed will be based on information gain.

```
In [6]: # Predicting result for test set
        y_pred = classifier.predict(X_test)
```

Here we are creating an array and storing the results of X_test dataset as predicted by our classifier.

Harshit Mishra(19BCE0799)

```
In [7]: # Printing the accuracy of our model
        from sklearn.metrics import accuracy_score
        accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)

Out[7]: 0.937442502299908

In [8]: # Printing the classification error
        1-accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)

Out[8]: 0.062557497700092

In [9]: # Printing the Sensitivity
        from sklearn.metrics import recall_score
        recall_score(y_test, y_pred)

Out[9]: 0.9921787709497206

In [10]: # Printing the Precision
         from sklearn.metrics import precision_score
         precision_score(y_test, y_pred)

Out[10]: 0.9357218124341412

In [12]: # Making confusion Matrix
         from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test, y_pred)

Out[12]: array([[131,  61],
                [  7, 888]], dtype=int64)
```

Here we have printed all the required results of accuracy, classification error, sensitivity, precision and confusion matrix.

Harshit Mishra(19BCE0799)

Result and Conclusion:

- Our confusion matrix is:

131	61
7	888

Thus,
True Positives (TP) = 888
True Negatives (TN) = 131
False Positives (FP) = 61
False Negatives (FN) = 7

- Accuracy of Model = (TP + TN) / (TP + TN + FP + FN)
  = (888 + 131) / (131 + 888 + 61 + 7)
  = 1019 / 1087
  = 0.9374 (same as Out[7])

- Classification Error = 1 – Accuracy
  = 1 – 0.9374
  = 0.06257 (same as Out[8])

- Sensitivity = TP / (TP + FN)
  = 888 / (888 + 7)
  = 888 / 895
  = 0.99217 (same as Out[9])

- Specificity = TN / (TN + FP)
  = 131 / (131 + 61)
  = 131 / 192
  = 0.68229

- Precision = TP / (TP + FP)
  = 888 / (888 + 61)
  = 888 / 949
  = 0.93572 (same as Out[10])

- Final results:

  Accuracy = 0.9374
  Classification Error = 0.0625
  Sensitivity = 0.9921
  Specificity = 0.6822
  Precision = 0.9357

Harshit Mishra(19BCE0799)

LAB ASSIGNMENT 3

Ques: Construct the Linear Regression Plot of Covid Cases.

Dataset Used: https://www.kaggle.com/imdevskp/covid19-corona-virus-india-dataset?select=nation_level_daily.csv

Procedure:

- We first import the dataset into our workspace using pandas.
- Then we need to encode our date attribute to date time stamp that can be used in our regression model.
- We then have to decide on the independent and dependent attributed to be used in our regression model.
- Next, we have to split our dataset into training set and test set.
- We then have to initialize our Linear regression model and fit it to the X_train and y_train.
- Next, we have to create another variable to store the results of X_test as predicted by our regression model.
- We then can find the scatter plot of our training sets and the best fit Regression line.
- We also can find the scatter plot of our test set and the best fit line of the training set (This would be same as best fit line of test set).
- Finally, we calculate our evaluation metrics to check the accuracy of our model.

Harshit Mishra(19BCE0799)

Code:

```
# Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# Importing the dataset
import datetime as dt
data = pd.read_csv("nation_level_daily.csv")
data['Date'] = pd.to_datetime(data['Date'], format = "%d %B ", errors='coerce')
data['Date'] = data['Date'].map(dt.datetime.toordinal)


# Creating the Independent and Dependent Variable
X = data.iloc[62:92, 0].values
y = np.asarray(data.iloc[62:92, 1].values)


# Splitting the datset into Training and Test Set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


# Training the Linear Regression Model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train.reshape(-1,1), y_train)


# Predicting the results
y_pred = regressor.predict(X_test.reshape(-1,1))


# Visualisng the training results
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, regressor.predict(X_train.reshape(-1,1)), color='blue')
plt.title('Covid Timeline in April')
plt.xlabel('Dates')
plt.ylabel('Number of cases')
plt.show()
```

Harshit Mishra(19BCE0799)

```
# Visualisng the test results
plt.scatter(X_test, y_test, color='red')
plt.plot(X_test, y_pred, color='blue')
plt.title('Covid Timeline in April')
plt.xlabel('Dates')
plt.ylabel('Number of cases')
plt.show()


# Printing Mean Absolute Error
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)


# Printing Mean Squared Error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)


# Printing Root Mean Squared Error
np.sqrt(mean_squared_error(y_test, y_pred))


# Printing Root Mean Squared Log Error
np.log(np.sqrt(mean_squared_error(y_test, y_pred)))


# Printing R Square
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Harshit Mishra(19BCE0799)

Code Snippet and Explanation:

```
In [1]: # Importing the libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

Here we are importing the necessary libraries.

```
In [2]: # Importing the dataset
        import datetime as dt
        data = pd.read_csv("nation_level_daily.csv")
        data['Date'] = pd.to_datetime(data['Date'], format = "%d %B ", errors='coerce')
        data['Date'] = data['Date'].map(dt.datetime.toordinal)
```

Since the date attribute in our dataset is not of the date_time_stamp and we cannot use it in our model, we need to encode it to the required format in order to use it in our regression model.

```
In [3]: # Creating the Independent and Dependent Variable
        X = data.iloc[62:92, 0].values
        y = np.asarray(data.iloc[62:92, 1].values)
```

Here, instead of considering all the dates, we are considering the number of covid cases only in the month of April.

```
In [4]: # Splitting the datset into Training and Test Set
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Here, we split our dataset with around 80% of data in training set and 20% of the data in test set.

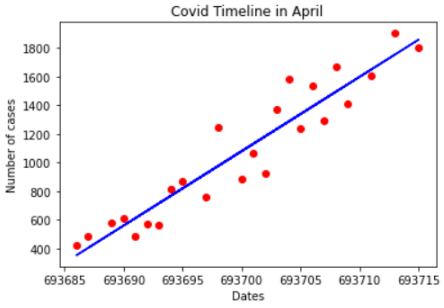
```
In [5]: # Training the Linear Regression Model
        from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
        regressor.fit(X_train.reshape(-1,1), y_train)

Out[5]: LinearRegression()
```

Here we have trained our Linear regression model with training dataset.

```
In [6]: # Predicting the results
        y_pred = regressor.predict(X_test.reshape(-1,1))
```

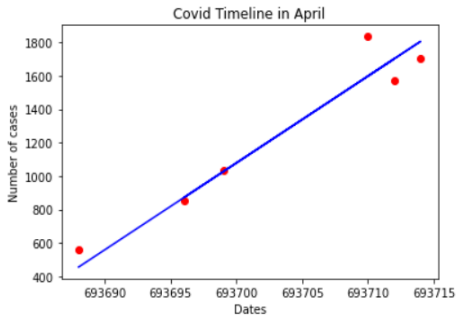
Here we are creating an array and storing the results of X_test dataset as predicted by our regressor.

```
In [7]: # Visualisng the training results
        plt.scatter(X_train, y_train, color='red')
        plt.plot(X_train, regressor.predict(X_train.reshape(-1,1)), color='blue')
        plt.title('Covid Timeline in April')
        plt.xlabel('Dates')
        plt.ylabel('Number of cases')
        plt.show()
```



Here, we are plotting the training sets with the best fit regression line. The dates are encoded in form of numbers and thus we have numbers such as 693700 and 693715 instead of 25th April or 30th April.

```
In [8]: # Visualisng the test results
        plt.scatter(X_test, y_test, color='red')
        plt.plot(X_test, y_pred, color='blue')
        plt.title('Covid Timeline in April')
        plt.xlabel('Dates')
        plt.ylabel('Number of cases')
        plt.show()
```



Here we have plotted our test set result with the regression line. Again, here we have dates in encoded format instead of the conventional date format.

Printing the evaluation metrics:

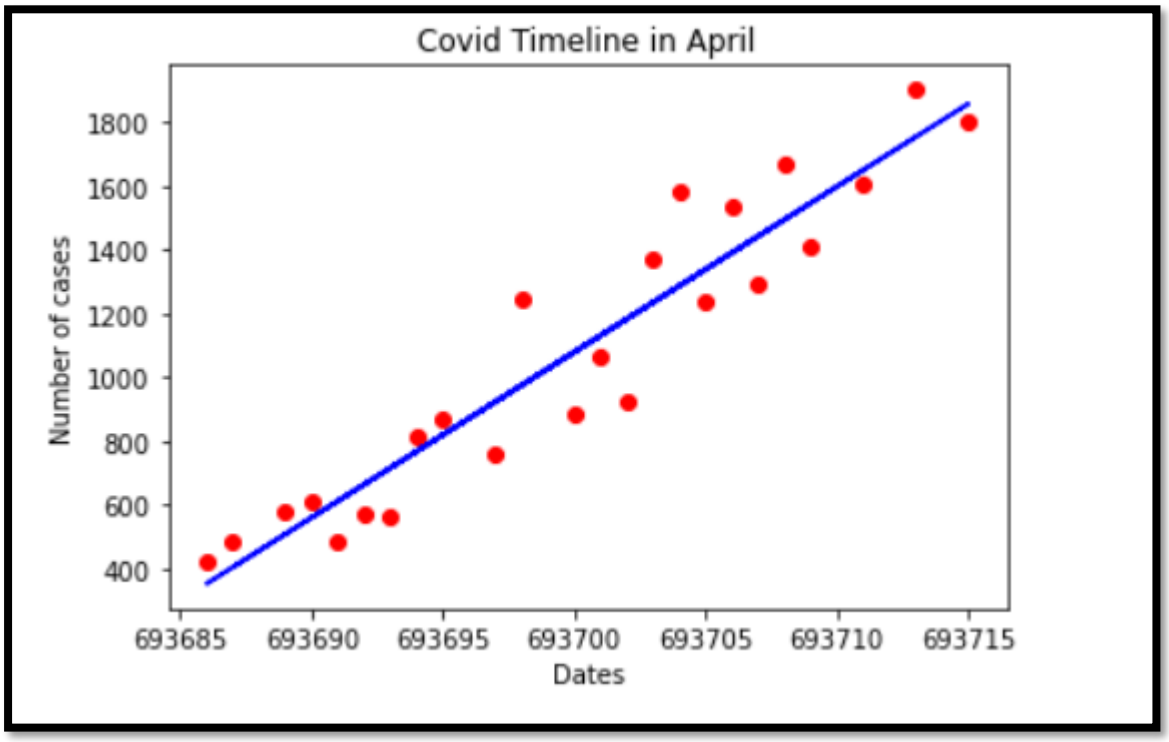
```
In [9]: # Printing Mean Absolute Error
        from sklearn.metrics import mean_absolute_error
        mean_absolute_error(y_test, y_pred)
Out[9]: 99.65776209781568

In [10]: # Printing Mean Squared Error
         from sklearn.metrics import mean_squared_error
         mean_squared_error(y_test, y_pred)
Out[10]: 15929.36025026464

In [11]: # Printing Root Mean Squared Error
         np.sqrt(mean_squared_error(y_test, y_pred))
Out[11]: 126.21156939942011

In [12]: # Printing Root Mean Squared Log Error
         np.log(np.sqrt(mean_squared_error(y_test, y_pred)))
Out[12]: 4.83795962101947

In [13]: # Printing R Square
         from sklearn.metrics import r2_score
         r2_score(y_test, y_pred)
Out[13]: 0.9281804971422801
```

Harshit Mishra(19BCE0799)

Result and Conclusion:

- Mean Absolute Error (MAE) = 99.65776
- Mean Squared Error (MSE) = 15929.36025
- Root Mean Squared Error (RMSE) = 126.21156
- Root Mean Squared Log Error (RMSLE) = 4.837959
- R Square Value (R^2) = 0.92818
  This suggests that there is a strong corelation between the days in April and increase in the number of covid19 cases.

- Training set Plot:



Harshit Mishra(19BCE0799)

- Test set plot: