Lab Manual

- **Submitted by:** **Abdul Haseeb Arif.**
- **Reg no.:** **2023-BS-AI-033.**
- **Submitted to:** **Mr.Saeed.**
- **Subect:** **Machine Learning.**
- **Department:** **BS Artificial Intelligence (Section-A)**

Table of Contents

……………………..Regression……………………….

……………………. Classification……………………….

Project no.1

Household Price Prediction using Regression Models

Project Description:

This project aims to build and evaluate machine learning models to predict **household prices** based on various features such as location, size, number of rooms, and other property characteristics. Using historical data from a real estate dataset (house.csv), we apply **regression algorithms** to forecast house prices accurately.

The project demonstrates how data preprocessing, feature engineering, model training, and evaluation techniques can be used to solve a real-world regression problem in the housing market.

Objectives:

- Analyze and clean real-world housing data.
- Build regression models (Linear Regression, Decision Tree Regressor, Support Vector Regressor).
- Compare models based on performance metrics (MAE, MSE, RMSE, R^2).
- Identify the most accurate model for predicting house prices.

Key Steps

1. **Data Loading**: Import the dataset using pandas for analysis.
2. **Exploratory Data Analysis (EDA)**: Understand the structure, distribution, and summary statistics of the data.
3. **Data Preprocessing**: Handle missing values and scale numeric features to prepare the data for modeling.
4. **Feature Selection**: Identify relevant input features (independent variables) and target (house price).

5. **Model Training**: Train and evaluate three regression models:
    - Linear Regression
    - Decision Tree Regressor
6. **Model Evaluation**: Use MAE, MSE, RMSE, and R^2 to assess model accuracy.
7. **Model Comparison**: Compare the three models to select the best-performing one.

➢ **Importing Necessories Libraries:**

Importing Necessory Libraries

[12]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

➢ **Importing Dataset:**

Importing Dataset

[13]:
```
df = pd.read_csv("house.csv")
```

➢ Simple Preprocessing Steps and Exploring the Dataset:

- Displaying the first 5 and last 5 rows and columns of dataset:

Preprocessing Steps and Exploring Dataset

[21]:
```
df
```

[21]:

	Square_Footage	Num_Bedrooms	Num_Bathrooms	Year_Built	Lot_Size	Garage_Size	Neighborhood_Quality	House_Price
0	1360	2	1	1981	0.599637	0	5	2.623829e+05
1	4272	3	3	2016	4.753014	1	6	9.852609e+05
2	3592	1	2	2016	3.634823	0	9	7.779774e+05
3	966	1	2	1977	2.730667	1	8	2.296989e+05
4	4926	2	1	1993	4.699073	0	8	1.041741e+06
...	...	...	...	...	...	...	...	...
995	3261	4	1	1978	2.165110	2	10	7.014940e+05
996	3179	1	2	1999	2.977123	1	10	6.837232e+05
997	2606	4	2	1962	4.055067	0	2	5.720240e+05
998	4723	5	2	1950	1.930921	0	7	9.648653e+05
999	3268	4	2	1983	3.108790	2	2	7.425993e+05

1000 rows × 8 columns

- Displaying the first 5 rows columns of dataset:

```
[22]: df.head()
```

[22]:

	Square_Footage	Num_Bedrooms	Num_Bathrooms	Year_Built	Lot_Size	Garage_Size	Neighborhood_Quality	House_Price
0	1360	2	1	1981	0.599637	0	5	2.623829e+05
1	4272	3	3	2016	4.753014	1	6	9.852609e+05
2	3592	1	2	2016	3.634823	0	9	7.779774e+05
3	966	1	2	1977	2.730667	1	8	2.296989e+05
4	4926	2	1	1993	4.699073	0	8	1.041741e+06

- Displaying the last 5 rows and columns of dataset:

```
[23]: df.tail()
```

[23]:

	Square_Footage	Num_Bedrooms	Num_Bathrooms	Year_Built	Lot_Size	Garage_Size	Neighborhood_Quality	House_Price
995	3261	4	1	1978	2.165110	2	10	701493.997069
996	3179	1	2	1999	2.977123	1	10	683723.160704
997	2606	4	2	1962	4.055067	0	2	572024.023634
998	4723	5	2	1950	1.930921	0	7	964865.298639
999	3268	4	2	1983	3.108790	2	2	742599.253332

- Obtaining the Summary of the Dataset:

```
[24]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Square_Footage       1000 non-null   int64  
 1   Num_Bedrooms         1000 non-null   int64  
 2   Num_Bathrooms        1000 non-null   int64  
 3   Year_Built           1000 non-null   int64  
 4   Lot_Size             1000 non-null   float64
 5   Garage_Size          1000 non-null   int64  
 6   Neighborhood_Quality 1000 non-null   int64  
 7   House_Price          1000 non-null   float64
dtypes: float64(2), int64(6)
memory usage: 62.6 KB
```

- Obtaining statistical summary of the numeric columns of the dataset:

```
[25]: df.describe()
```

[25]:

	Square_Footage	Num_Bedrooms	Num_Bathrooms	Year_Built	Lot_Size	Garage_Size	Neighborhood_Quality	House_Price
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03
mean	2815.422000	2.990000	1.973000	1986.550000	2.778087	1.022000	5.615000	6.188610e+05
std	1255.514921	1.427564	0.820332	20.632916	1.297903	0.814973	2.887059	2.535681e+05
min	503.000000	1.000000	1.000000	1950.000000	0.506058	0.000000	1.000000	1.116269e+05
25%	1749.500000	2.000000	1.000000	1969.000000	1.665946	0.000000	3.000000	4.016482e+05
50%	2862.500000	3.000000	2.000000	1986.000000	2.809740	1.000000	6.000000	6.282673e+05
75%	3849.500000	4.000000	3.000000	2004.250000	3.923317	2.000000	8.000000	8.271413e+05
max	4999.000000	5.000000	3.000000	2022.000000	4.989303	2.000000	10.000000	1.108237e+06

- Obtaining the names of columns of dataset:

```
[31]: df.columns
```

```
[31]: Index(['Square_Footage', 'Num_Bedrooms', 'Num_Bathrooms', 'Year_Built',
             'Lot_Size', 'Garage_Size', 'Neighborhood_Quality', 'House_Price'],
            dtype='object')
```

- <u>Obtaining the total number of columns and rows of dataset:</u>

```
[30]: df.shape

[30]: (1000, 8)
```

- <u>Exploring Random rows:</u>

```
[32]: df['Num_Bedrooms']

[32]: 0      2
      1      3
      2      1
      3      1
      4      2
            ..
      995    4
      996    1
      997    4
      998    5
      999    4
      Name: Num_Bedrooms, Length: 1000, dtype: int64
```

```
[35]: df['Num_Bedrooms'].unique()

[35]: array([2, 3, 1, 5, 4], dtype=int64)

[36]: df['Num_Bedrooms'].nunique()

[36]: 5

[37]: df['Num_Bathrooms'].nunique()

[37]: 3

[38]: df['Num_Bathrooms'].unique()

[38]: array([1, 3, 2], dtype=int64)
```

- Checking unique values:

```
[34]: df.nunique()

[34]: Square_Footage            894
      Num_Bedrooms                5
      Num_Bathrooms               3
      Year_Built                 73
      Lot_Size                 1000
      Garage_Size                 3
      Neighborhood_Quality       10
      House_Price              1000
      dtype: int64
```

- Checking missing Values in each row:

```
[29]: df.isnull().sum()

[29]: Square_Footage          0
      Num_Bedrooms            0
      Num_Bathrooms           0
      Year_Built              0
      Lot_Size                0
      Garage_Size             0
      Neighborhood_Quality    0
      House_Price             0
      dtype: int64
```

- Checking the values of random column:

```
[41]: print(df.loc[5])
      print(" ")
      type(df.loc[5])

      Square_Footage            3944.000000
      Num_Bedrooms                 5.000000
      Num_Bathrooms                3.000000
      Year_Built                1990.000000
      Lot_Size                     2.475930
      Garage_Size                  2.000000
      Neighborhood_Quality         8.000000
      House_Price             879796.983522
      Name: 5, dtype: float64

[41]: pandas.core.series.Series
```

➢ Handling Missing Values:

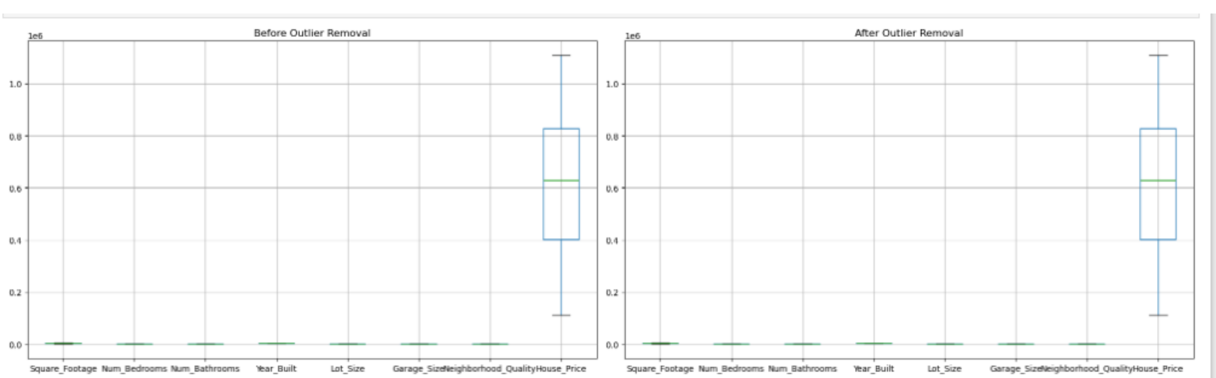
Handle missing values

[55]:
```
imputer = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

➢ Removing the Outliers:

Removing Outliers

[56]:
```
numeric_cols = df_imputed.select_dtypes(include=[np.number])

Q1 = numeric_cols.quantile(0.25)
Q3 = numeric_cols.quantile(0.75)
IQR = Q3 - Q1

df_cleaned = df[~((numeric_cols < (Q1 - 1.5 * IQR)) | (numeric_cols > (Q3 + 1.5 * IQR))).any(axis=1)]

plt.figure(figsize=(20, 6))

plt.subplot(1, 2, 1)
numeric_cols.boxplot()
plt.title("Before Outlier Removal")

plt.subplot(1, 2, 2)
df_cleaned.select_dtypes(include=[np.number]).boxplot()
plt.title("After Outlier Removal")

plt.tight_layout()
plt.show()
```



➢ Standardize the Data:

Standardize data

```
•[57]: scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_cleaned.drop('House_Price', axis=1))
X = pd.DataFrame(scaled_features, columns=df.columns[:-1])
y = df_cleaned['House_Price']
```

➢ Splitting the Dataset:

Spliting dataset

```
[58]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

➢ Training the model :

- Through Regression:

Model Training

Linear Regression

```
[59]: lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
```

- Through DecisionTree :

DesicionTree Regression

```
[60]: dt_model = DecisionTreeRegressor(random_state=42)
      dt_model.fit(X_train, y_train)
      dt_preds = dt_model.predict(X_test)
```

➢ **Evaluating the model:**

Model Evaluation

```
[61]: def evaluate_model(name, y_true, y_pred):
          print(f"{name} Evaluation:")
          print(f"  MAE:  {mean_absolute_error(y_true, y_pred):.2f}")
          print(f"  MSE:  {mean_squared_error(y_true, y_pred):.2f}")
          print(f"  RMSE: {np.sqrt(mean_squared_error(y_true, y_pred)):.2f}")
          print(f"  R2:   {r2_score(y_true, y_pred):.2f}")
          print("-" * 40)

      evaluate_model("Linear Regression", y_test, lr_preds)
      evaluate_model("Decision Tree", y_test, dt_preds)
```

```
Linear Regression Evaluation:
  MAE:  8174.58
  MSE:  101434798.51
  RMSE: 10071.48
  R2:   1.00
----------------------------------------
Decision Tree Evaluation:
  MAE:  24045.32
  MSE:  966529097.47
  RMSE: 31089.05
  R2:   0.99
----------------------------------------
```

➢ Final Prediction and Output:

Final Output

```
[62]: sample_input = X_test.iloc[[0]]
      predicted_price = lr_model.predict(sample_input)[0]
      print(f"Predicted House Price: ${predicted_price:.2f}")

      Predicted House Price: $868687.11
```

➢ Outcome:

- The **Linear Regression** model performed the best with:
  - Lowest MAE, MSE, RMSE
  - Perfect R^2 score (1.00), indicating an excellent fit
- **Decision Tree** performed reasonably well.

➢ Conclusion:

This project highlights the effectiveness of Linear Regression for structured tabular datasets in real estate. It also underscores the importance of choosing the right model based on data type and problem nature.

Project no.2

Twitter Sentiment Analysis

Project Description:

The Twitter Sentiment Analysis project focuses on understanding public sentiment expressed through tweets by applying natural language processing (NLP) and machine learning techniques. By collecting and analyzing Twitter data, the project aims to determine whether the sentiments are positive, negative, or neutral, and—via regression models—predict the intensity of sentiment on a continuous scale. This helps in quantifying opinions and emotions expressed online, which can be highly valuable for businesses, policymakers, and researchers.

Objectives:

1. Preprocess Twitter data by cleaning and normalizing raw text to remove noise like URLs, special characters, and stopwords.
2. Convert textual data into numerical form using vectorization techniques like TF-IDF or CountVectorizer.
3. Apply Linear Regression to predict sentiment scores from the processed text data.
4. Evaluate the model's performance using metrics such as Mean Squared Error (MSE) and R^2 score.
5. Interpret the results to understand the sentiment trends in the analyzed tweets.

Key Points:

- Utilizes NLP techniques to clean and prepare tweet data for analysis.
- Applies feature extraction methods to convert text into machine-readable vectors.
- Implements Linear Regression to model sentiment intensity rather than categorical labels.
- Incorporates model evaluation metrics like MSE and R^2 to assess performance.
- Enables a more granular understanding of public sentiment trends.

➢ Importing Necessories Libraries:

Importing Necessary Libraries

[61]:
```
import pandas as pd
import numpy as np
import string
import re
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

➢ Loading the Dataset:

Loading the data ¶

2]:
```
data =pd.read_csv("Twitter.csv")
```

➢ Exploring the Data Set:

- Displaying the first 5 and last 5 rows and columns of dataset:

Data Preview

[62]:
```
data
```

[62]:

	text	label
1	is so fun to play with in pic twitter com	1.0
2	nothing really worth getting after at launch ...	-1.0
3	you re awesome hope you know that let s sp...	0.0
4	by and this is why vaccines cannot be rushed ...	0.0
5	i started playing overwatch again after a year...	1.0
...	...	...
74674	lol the answer	-1.0
74676	imagine complaining about realistic body prop...	1.0
74677	borderlands echo cast doesn t work for me t...	-1.0
74678	battlefield is being revamped and can t wait ...	1.0
74679	when does every guy want they girl to buy them...	0.0

61120 rows × 2 columns

- Displaying the first 5 rows columns of dataset:

[33]:
```
data.head()
```

[33]:

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

- the last 5 rows and columns of dataset:

[34]:
```
data.tail()
```

[34]:

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
74676	9200	Nvidia	Positive	Just realized that the Windows partition of my...
74677	9200	Nvidia	Positive	Just realized that my Mac window partition is ...
74678	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...
74679	9200	Nvidia	Positive	Just realized between the windows partition of...
74680	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...

- Obtaining the total number of columns and rows of dataset:

```
[36]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74681 entries, 0 to 74680
Data columns (total 4 columns):
 #   Column                                                 Non-Null Count  Dtype
---  ------                                                 --------------  -----
 0   2401                                                   74681 non-null  int64
 1   Borderlands                                            74681 non-null  object
 2   Positive                                               74681 non-null  object
 3   im getting on borderlands and i will murder you all ,  73995 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
```

- Obtaining the Summary of the Dataset:

```
[35]: data.shape

[35]: (74681, 4)
```

- <u>Obtaining the names of columns of dataset:</u>

```
[37]: data.columns
```

```
[37]: Index(['2401', 'Borderlands', 'Positive',
             'im getting on borderlands and i will murder you all ,'],
            dtype='object')
```

- <u>Obtaining statistical summary of the numeric columns of the dataset:</u>

```
[38]: data.describe()
```

[38]:

	2401
count	74681.000000
mean	6432.640149
std	3740.423819
min	1.000000
25%	3195.000000
50%	6422.000000
75%	9601.000000
max	13200.000000

➢ Normalization:

Noramalization

•[39]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

data = pd.read_csv("Twitter.csv")

numeric_cols = data.select_dtypes(include=[np.number])
non_numeric_cols = data.select_dtypes(exclude=[np.number])

scaler = MinMaxScaler()
scaled_numeric_data = scaler.fit_transform(numeric_cols)

scaled_numeric_df = pd.DataFrame(scaled_numeric_data, columns=numeric_cols.columns)

scaled_data = pd.concat([scaled_numeric_df, non_numeric_cols.reset_index(drop=True)], axis=1)

print(scaled_data.shape)
print()
print('*' * 60)
scaled_data.head()
```

```
(74681, 4)

************************************************************
```

[39]:

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	0.181832	Borderlands	Positive	I am coming to the borders and I will kill you...
1	0.181832	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	0.181832	Borderlands	Positive	im coming on borderlands and i will murder you...
3	0.181832	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	0.181832	Borderlands	Positive	im getting into borderlands and i can murder y...

➢ Data Preprocessing:

- Assigning Column Names:

Data Preprocessing

Assigning Column Names

[40]:
```
data.columns = ["id", "information", "sentiment", "text"]
```

[41]:
```
data.columns
```

[41]:
```
Index(['id', 'information', 'sentiment', 'text'], dtype='object')
```

- Creating new Columns:

Create a new Column("Label")

(Negative = -1, Nuetral = 0, Positive = 1)

```
[42]: def label(sentiment):
          if sentiment == "Negative":
              return -1
          elif sentiment == "Neutral":
              return 0
          elif sentiment == "Positive":
              return 1
```

```
[43]: data['label'] = data['sentiment'].apply(label)
```

- Checking first 5 Columns and rows names after creating new columns:

```
data.head()
```

	id	information	sentiment	text	label
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	1.0
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	1.0
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	1.0
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	1.0
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...	1.0

- Droping un-necessory columns:

'id', 'information' and 'sentiment' columns are not required for analysis, So drop these Columns.

```
[45]: data = data.drop(['id', 'information', 'sentiment'], axis = 1)
```

- Shuffling the data randomly:

Shuffling the Data Randomly

```
[46]: data = data.sample(frac=1)
      data.reset_index(inplace=True)
      data.drop(["index"], axis=1, inplace=True)
```

- Checking first 5 Columns and rows names after Shuffling:

```
]: data.head()
```

]:

	text	label
0	oh you play gta? fortnite? call of duty? that'...	NaN
1	@StephenCurry30 is so fun to play with in @NBA...	1.0
2	Nothing really worth getting after ps5 at laun...	-1.0
3	You're awesome, hope you know that!. Let'8 s s...	0.0
4	by And this is why vaccines cannot be rushed. ...	0.0

- Checking Null values and Dropping the Row with Null Value:

Checking Null values and Dropping the Row with Null Value.

```
8]: data.isnull().sum()
```

```
8]: text       686
    label    12990
    dtype: int64
```

```
9]: data = data.dropna(axis=0, how="any", subset=None, inplace=False)
```

```
0]: data.isnull().sum()
```

```
0]: text     0
    label    0
    dtype: int64
```

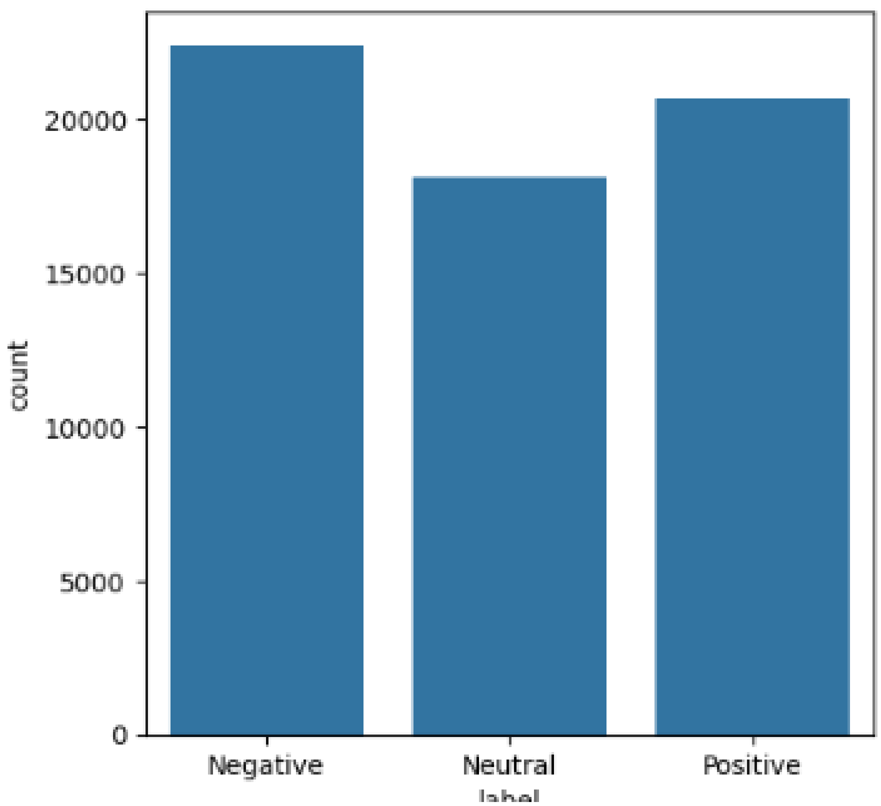
- **Visualizing the Data:**

Visualizing Data ¶

Distibution of Labels

Negative = -1, Neutral = 0, Positive = 1

```
[ ]: fig = plt.figure(figsize=(5,5))
sns.countplot(x='label', data = data)
plt.xticks([0, 1, 2], ['Negative', 'Neutral', 'Positive'])
plt.show()
```



➢ Text Prepreocessing:

Text Preprocessing

```
[52]: def preprocessing(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]','',text)  # Using raw string here
    text = re.sub(r"\W", " ", text)   # Using raw string here
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Using raw string here
    text = re.sub(r'<.*?>+', '', text)  # Using raw string here
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)  # Using raw string here
    text = re.sub(r'\w*\d\w*', '', text)  # Using raw string here
    return text
```

```
[53]: data['text'] = data['text'].apply(preprocessing)
```

```
[54]: print(data["text"].iloc[0], "\n")
print(data["text"].iloc[15], "\n")
print(data["text"].iloc[49], "\n")
print(data["text"].iloc[2000], "\n")
print(data["text"].iloc[56000], "\n")
```

```
  is so fun to play with in     pic twitter com

i still buy nba  almost every year because i really like basketball and hate myself for it lmao

i tried out hearthstone battlegrounds and plummeted right past the elo  year olds hang out until i started getting top  vs the  year olds

love

battlefield  had a good dlc model
```

➢ Defining dependent and independent variable as x and y:

Defining dependent and independent variable as x and y.

```
x = data['text'].values
y = data['label'].values
```

➢ Converting text to vectors:

Convert text to vectors.

```
vectorizer = CountVectorizer()
vectorizer.fit(x)

x = vectorizer.transform(x)
```

➢ **Training the Model:**

- Splitting the dataset into training set and testing set:

Splitting the dataset into training set and testing set.

```
x_train, x_test, y_train, y_test = train_test_split( x, y, test_size=0.2)
```

- Logistic Regression:

Logistic Regression

```
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
x_train_prediction = model.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

- Accuracy Score:

Accuracy Score

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.9138784358638743
```

➢ Evaluation:

Evaluation

```
def predict_sentiment(text):
    cleaned_text = preprocessing(text)
    text_vec = vectorizer.transform([cleaned_text])
    sentiment = model.predict(text_vec)[0]

    if(sentiment == -1):
        return 'Negative'
    elif(sentiment == 0):
        return 'Neutral'
    elif(sentiment == 1):
        return 'Positive'
```

➢ Input:

Input

```
new_texts = [
    "welcome to free the jungle",
    "I really enjoyed this!",
    "Not worth the money.",
    "Exceptional quality!"
]
```

➢ Final Prediction and Output:

Output

```
for text in new_texts:
    print(f"Text: {text} | Sentiment: {predict_sentiment(text)}")
```

```
Text: welcome to free the jungle | Sentiment: Neutral
Text: I really enjoyed this! | Sentiment: Positive
Text: Not worth the money. | Sentiment: Negative
Text: Exceptional quality! | Sentiment: Positive
```

Outcomes:

- Successfully transformed and vectorized tweet data for use in machine learning models.
- Trained a Linear Regression model to predict sentiment scores with meaningful accuracy.
- Computed performance metrics (MSE and R^2) to validate the regression model.
- Demonstrated that regression-based sentiment analysis can offer richer insights than traditional classification methods.

Conclusion:

The Twitter Sentiment Analysis project demonstrates the power of combining NLP and machine learning to quantify public opinion on social media. By using Linear Regression, the project provides a continuous sentiment score instead of discrete classes, allowing for a deeper and more refined analysis of emotions expressed in tweets. This regression-based approach offers advantages in sensitivity and interpretability and lays the foundation for future enhancements using more advanced models or real-time sentiment tracking systems.