**Documentation for ML Model on Stock Prediction**

**Naman Kumar Singh**

**Corizo Batch June-July (Machine Learning)**

**Title**: **Stock Price Prediction using Linear Regression**

**Objective**: To build a linear regression model to predict the closing prices of Tesla stock.

**1. Import Libraries**

python

Copy code

import pandas as pd

import numpy as np

from sklearn import metrics

import matplotlib.pyplot as plt

from sklearn.linear\_model import SGDRegressor

from sklearn.preprocessing import StandardScaler

%matplotlib inline

**Description**:

* **Pandas** for data manipulation.
* **NumPy** for numerical computations.
* **Metrics** for evaluating the model.
* **Matplotlib** for plotting graphs.
* **SGDRegressor** for linear regression.
* **StandardScaler** for scaling data.

**2. Load Data**

python

Copy code

df = pd.read\_csv("tesla.csv")

df.head()

**Description**:

* Load the Tesla stock data from a CSV file.
* Display the first few rows to understand the data structure.

**3. Time Format**

python

Copy code

df['Date'] = pd.to\_datetime(df['Date'], format='%d-%m-%Y')

df['Date\_ordinal'] = df['Date'].map(pd.Timestamp.toordinal)

df.head()

**Description**:

* Convert the 'Date' column to datetime format.
* Create a new column 'Date\_ordinal' with ordinal values of dates for regression purposes.

**4. Data Exploration**

python

Copy code

# Check the number of rows and columns

df.shape

# Checking for null entries

df.isnull().sum()

# Drop any null values

df.dropna(inplace=True)

df

# Data info

df.info()

**Description**:

* Check the dimensions of the dataset.
* Check for any null values and drop them if present.
* Display the info of the dataset to understand its structure.

**5. Plot the Closing Prices**

python

Copy code

# Plot the closing prices

df['Close'].plot(figsize=(10,8))

plt.xlabel('Date')

plt.ylabel('Close Price')

plt.title('Closing Prices of Tesla Stock')

plt.show()

**Description**:

* Plot the closing prices to visualize the trend over time.

**6. Feature Selection**

python

Copy code

# Define features and target variable

x = df[['Date\_ordinal', 'Open', 'High', 'Low', 'Volume']]

y = df['Close']

**Description**:

* Select the relevant features for the model.
* Define the target variable 'Close' which we want to predict.

**7. Train-Test Split**

python

Copy code

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0)

print(x\_train.shape, x\_test.shape)

**Description**:

* Split the data into training and testing sets.
* Print the shapes of the training and testing sets.

**8. Linear Regression Model**

python

Copy code

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import confusion\_matrix, accuracy\_score

# Initialize and train the model

regressor = LinearRegression()

regressor.fit(x\_train, y\_train)

# Print weights and biases

print(regressor.coef\_)

print(regressor.intercept\_)

**Description**:

* Initialize the linear regression model.
* Train the model on the training data.
* Print the coefficients and intercept of the model.

**9. Predictions**

python

Copy code

# Predict the values

predicted = regressor.predict(x\_test)

print(x\_test)

\_df = pd.DataFrame({"Actual": y\_test, "Predicted": predicted})

print(\_df)

**Description**:

* Use the trained model to make predictions on the test data.
* Display the actual and predicted values side by side.

**10. Model Evaluation**

python

Copy code

# Evaluate the model

regressor.score(x\_test, y\_test)

import math

print("Mean Absolute Error", metrics.mean\_absolute\_error(y\_test, predicted))

print("Mean Square Error", metrics.mean\_squared\_error(y\_test, predicted))

print("Root Mean Error", math.sqrt(metrics.mean\_squared\_error(y\_test, predicted)))

**Description**:

* Evaluate the model using various metrics such as Mean Absolute Error, Mean Square Error, and Root Mean Error.

**11. Plot Actual vs Predicted Values**

python

Copy code

# Plot actual vs predicted values

plt.figure(figsize=(10, 6))

plt.plot(df['Date'], df['Close'], color='blue', label='Actual Values')

plt.plot(df['Date'], df['Close'], color='red', linestyle='dashed', label='Predicted Values')

plt.xlabel('Date')

plt.ylabel('Close Price')

plt.title('Actual vs Predicted Close Prices')

plt.legend()

plt.show()

**Description**:

* Plot the actual closing prices against the predicted closing prices to visualize the model's performance.

**Conclusion**

This documentation walks through the process of loading data, preprocessing it, training a linear regression model, making predictions, and evaluating the model's performance. This can serve as a basis for further refinement and development of more complex models for stock price prediction.