**SAMPLE CODE**

import numpy as np

import pandas as pd

#import quandl

import datetime

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('seaborn-darkgrid')

plt.rc('figure', figsize=(16,10))

plt.rc('lines', markersize=4)

# Set start and end date for stock prices

start\_date = datetime.date(2009, 3,8)

end\_date = datetime.date.today()

# Load data from Quandl

#data = quandl.get('FSE/SAP\_X', start\_date=start\_date, end\_date=end\_date)

# Save data to CSV file

data = pd.read\_csv("data/sap\_stock.csv", low\_memory = False, skiprows = 1, encoding = "ISO-8859-1")

#data.to\_csv('data/sap\_stock.csv')

print("The GTD dataset has {} samples with {} features.".format(\*data.shape))

data.head()

# Check data types in columns

data.info()

# Get descriptive statistics summary of data set

data.describe()

# Display features in data set

data.columns

#data.drop(['Date'], axis = 1)

# Create a new DataFrame with only closing price and date

df = pd.DataFrame(data, columns=['Date','Close'])

# Reset index column so that we have integers to represent time for later analysis

df = df.reset\_index()

df.head()

# Check data types in columns

df.info()

# Check for missing values in the columns

df.isna().values.any()

# Import matplotlib package for date plots

import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year

yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes

fig, ax = plt.subplots()

ax.plot(df['Date'], df['Close'])

# Format the ticks

ax.xaxis.set\_major\_locator(years)

ax.xaxis.set\_major\_formatter(yearsFmt)

# Set figure title

plt.title('Close Stock Price History [2009 - 2019]', fontsize=16)

# Set x label

plt.xlabel('Date', fontsize=14)

# Set y label

plt.ylabel('Closing Stock Price in $', fontsize=14)

# Rotate and align the x labels

fig.autofmt\_xdate()

# Show plot

plt.show()

# Import package for splitting data set

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

# Split data into train and test set: 80% / 20%

train, test = train\_test\_split(df, test\_size=0.20)

# Import package for linear model

from sklearn.linear\_model import LinearRegression

# Reshape index column to 2D array for .fit() method

X\_train = np.array(train.index).reshape(-1, 1)

y\_train = train['Close']

# Create LinearRegression Object

model = LinearRegression()

# Fit linear model using the train data set

model.fit(X\_train, y\_train)

# The coefficient

print('Slope: ', np.asscalar(np.squeeze(model.coef\_)))

# The Intercept

print('Intercept: ', model.intercept\_)

# Train set graph

plt.figure(1, figsize=(16,10))

plt.title('Linear Regression | Price vs Time')

plt.scatter(X\_train, y\_train, edgecolor='w', label='Actual Price')

plt.plot(X\_train, model.predict(X\_train), color='r', label='Predicted Price')

plt.xlabel('Integer Date')

plt.ylabel('Stock Price')

plt.legend()

plt.show()

# Create test arrays

X\_test = np.array(test.index).reshape(-1, 1)

y\_test = test['Close']

# Generate array with predicted values

y\_pred = model.predict(X\_test)

print(y\_pred[0:25])

#df['Prediction'] = y\_pred[:24]

print(type(y\_pred))

# Generate 25 random numbers

randints = np.random.randint(2550, size=25)

# Select row numbers == random numbers

df\_sample = df[df.index.isin(randints)]

df\_sample['Prediction']=y\_pred[0:25]

# Create subplots to plot graph and control axes

fig, ax = plt.subplots()

df\_sample.plot(x='Date', y=['Close', 'Prediction'], kind='bar', ax=ax)

# Set figure title

plt.title('Comparison Predicted vs Actual Price in Sample data selection', fontsize=16)

#

# Set x label

plt.xlabel('Date', fontsize=14)

# Set y label

plt.ylabel('Stock Price in $', fontsize=14)

# Show plot

plt.show()

plt.figure(1, figsize=(16,10))

plt.title('Linear Regression | Price vs Time Future')

plt.plot(X\_test, model.predict(X\_test), color='r', label='Predicted Price')

plt.scatter(X\_test, y\_test, edgecolor='w', label='Actual Price')

plt.xlabel('Integer Date')

plt.ylabel('Stock Price in $')

plt.show()

# Plot predicted vs actual prices

plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')

plt.title('Predicted Actual Price in future')

plt.show()

# Import norm package to plot normal distribution

from scipy.stats import norm

# Fit a normal distribution to the data:

mu, std = norm.fit(y\_test - y\_pred)

ax = sns.distplot((y\_test - y\_pred), label='Residual Histogram & Distribution')

# Calculate the pdf over a range of values

x = np.linspace(min(y\_test - y\_pred), max(y\_test - y\_pred), 100)

p = norm.pdf(x, mu, std)

# And plot on the same axes that seaborn put the histogram

ax.plot(x, p, 'r', lw=2, label='Normal Distribution')

plt.legend()

plt.show()

# Add new column for predictions to df

df['Prediction'] = model.predict(np.array(df.index).reshape(-1, 1))

# Statistical summary of test data

df['Close'].describe()

# Calculate and print values of MAE, MSE, RMSE

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

print('R2: ', metrics.r2\_score(y\_test, y\_pred))

from sklearn.metrics import explained\_variance\_score

explained\_variance\_score(y\_test, y\_pred)