PROJECT REPORT ON:- AI IN BUSINESS (STOCK MARKET AND PREDCTION).

****

****

ABSTRACT

AI IN STOCK MARKET

Stocks are possibly the most popular financial instrument invented for building wealth and are the centrepiece of any investment portfolio. The advance in trading technology has opened up the market so that nowadays nearly anybody can own stocks. From last few decades, there seen explosive increase in the average person's interest for the stock market. In the financially explosive market, as the stock market, it is important to have a very accurate prediction of a future trend. Because of the financial crisis and re-coding profits, it is compulsory to have

a secure prediction of the values of the stocks. Predicting a non-linear signal requires progressive algorithms of machine learning with help of Artificial Intelligence (AI).

In our research, we are going to use Machine Learning Algorithm specially focus on Linear Regression (LR) and Logistic Regression. Exponential Smoothing and Time Series Forecasting using Ms Excel as the best statistical tool for graph and tabular representation of prediction results. We obtain data from various source like from Google after implementation LR we successfully predict stock market trend for next month and also measured accuracy according to measurements.

INTRODUCTION

Stock market is trading platform where different investors sale and purchase shares according to stock avail ability. Stock market ups and downs effects the profit of stakeholders. If market prices going up with available stock then stakeholders get profit with their purchased stocks. In other case, if market going down with available stock prices then stakeholders have to face losses. Buyers buy stocks with low prices and sell stocks at high prices and try to get huge profit. Similarly, sellers sell their products at high prices for profit purpose. Stock market work as trusty platform among sellers and buyers. Advances in Artificial Intelligence (AI) supporting a lot in each field of life with its intelligent features. Several algorithms present in AI that performing their role in future predictions. Machine learning (ML) is a field of artificial intelligence (AI) that can be considered as we train machines with data and analysis future

with test data. Machines can be trained on the basis of some standard that are called algorithms. Stock market predictions can be great beneficial to businessman. Stock Market Preddiction provide future trend of stock prices on the basis of previous history. If stakeholders get future predictions then investment can lead him toward profit. Predictions can be 50% correct and 50% wrong as it is risk of business. Risks facing capability in business lead toward success. In any field of life, we take risks for success. Similarly, we rely on ML predictions about future prices of stock. . Before working on actual problem SMP, complete understanding of ML algorithms role in prediction is also necessary. That's why in this project we explained complete working scenario and problem. Several Machine learning algorithms can be used for stock market prediction but in this research we used few algorithms like Linear regression (LR) and Logistic regression.

REGRESSION

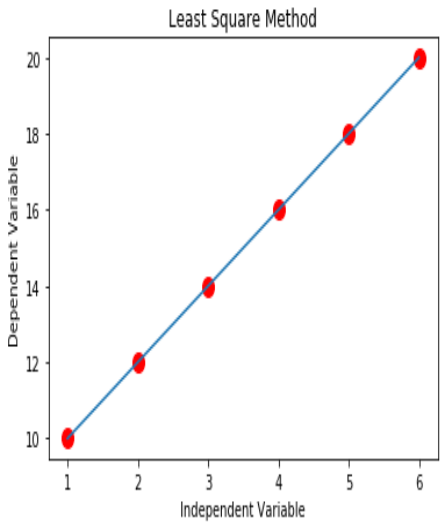
- **Supervised Learning.**
- **Output is a continuous quantity.**
- **Main aim to forecast or predict.**
- **Eg: Predict stock market price.**
- **Algorithm: Linear Regression.**

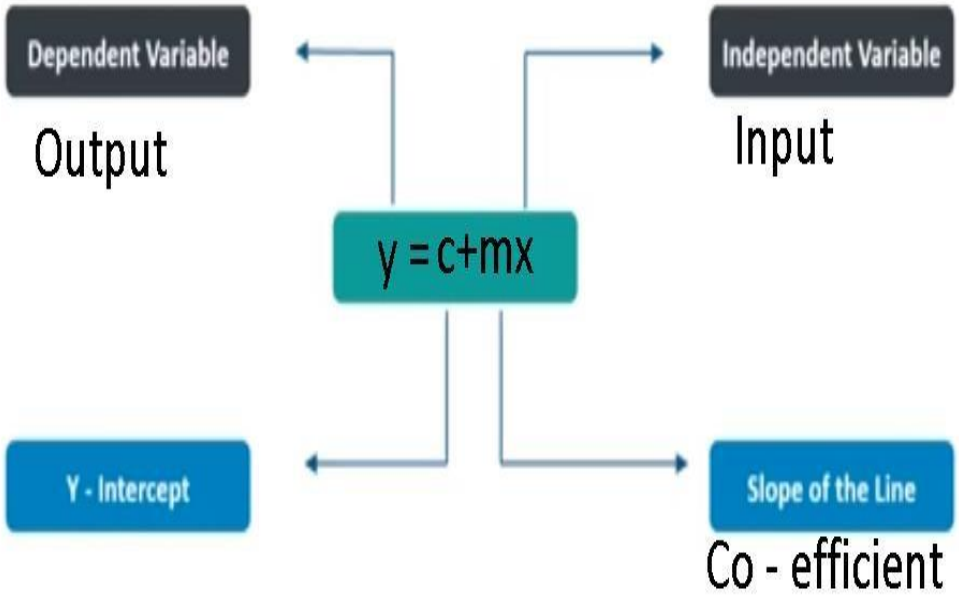
CLASSIFICATION

- **Supervised Learning.**
- **Output is a continuous quantity.**
- **Main aim to compute the category of the data**
- **Eg: Classify emails as spam and non-spam.**
- **Algorithm: Logistic Regression.**

REQUIRED FIELDS

- **Libraries**
  - ✓ **Numpy**
  - ✓ **Pandas**
  - ✓ **Matlpotlib.pyplot**
- **Linear Regression**
  - ✓ **Linear relationship between input(x) and output(y)**
  - ✓ **Equation of straight line [y = mx + c]**
    - **1. X- Input**
    - **2. Y- Output**
    - **3. M- Slope**
    - **4. C- Intercept**
  - ✓ **SIMPLE LINEAR REGRESSION (1 input column)**
  - ✓ **MULTI LINEAR REGRESSION (multiple input columns)**
  - ✓ **Method used – ordinary least square method**



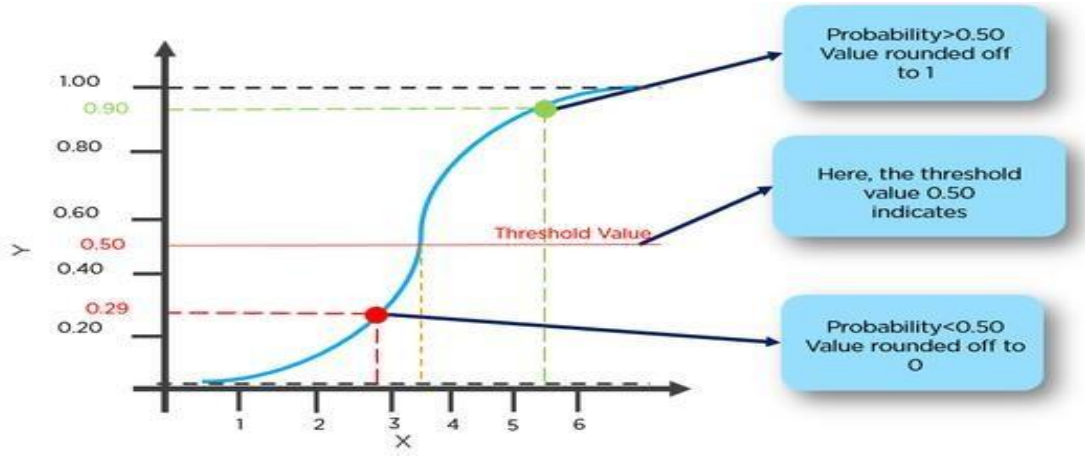


$$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

$\bar{x}$ - mean of x
$\bar{y}$ - mean of y

➢ Logistic Regression

- ✓ **In Statics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.**



$$y = f(x) = \frac{1}{1 + e^{-(x)}}$$

Tolls we use

- ✓ Modules
- ✓ NLP
- ✓ Google colab notebook
- ✓ Data from internet.
- ✓ Kaggle

Some key points about code:-

- **Using pandas and numpy.**
- **Using apple stock csv file.**
- **SVM(simple moving average)**- It is the unweighted mean of previous data. The mean is normally taken from an equal number of data on either side of a central value. This ensures that variations in the mean are aligned with the variations in the data rather than being shifted in time.
- **EMA(exponential moving average)**-It is a type of moving average that places a greater weight and significance on the most recent data points. It is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average (SMA), which applies an equal weight to all observations in the period.
- **rolling(window=30).mean()** is a pandas function that calculates the rolling average of the data. The **window** parameter specifies the size of the window over which the rolling average is calculated.
- The **R2_score** represents the proportion of variance in the dependent variable that is explained by the independent variable. The closer the score is to 1, the better the model fits the data. A score of 0 means that the model does not explain any of the variance in the dependent variable. However, note that the R-squared score is not the only metric to evaluate the

accuracy of a model and it's important to consider other metrics as well.

- **Matplotlib**:-For ploting the graphs.

CODE

```
[ ] import pandas as pd
    import numpy as np
```

```
df = pd.read_csv("/content/AAPL.csv")
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600
...	...	...	...	...	...	...	...
10463	2022-06-13	132.869995	135.199997	131.440002	131.880005	131.880005	122207100
10464	2022-06-14	133.130005	133.889999	131.479996	132.759995	132.759995	84784300
10465	2022-06-15	134.289993	137.339996	132.160004	135.429993	135.429993	91533000
10466	2022-06-16	132.080002	132.389999	129.039993	130.059998	130.059998	108123900
10467	2022-06-17	130.070007	133.080002	129.809998	131.559998	131.559998	134118500

10468 rows × 7 columns

```
df.info
```

```
<bound method DataFrame.info of             Date        Open        High         Low       Close   Adj Close  \
0      1980-12-12    0.128348    0.128906    0.128348    0.128348    0.100178
1      1980-12-15    0.122210    0.122210    0.121652    0.121652    0.094952
2      1980-12-16    0.113281    0.113281    0.112723    0.112723    0.087983
3      1980-12-17    0.115513    0.116071    0.115513    0.115513    0.090160
4      1980-12-18    0.118862    0.119420    0.118862    0.118862    0.092774
...           ...         ...         ...         ...         ...         ...
10463  2022-06-13  132.869995  135.199997  131.440002  131.880005  131.880005
10464  2022-06-14  133.130005  133.889999  131.479996  132.759995  132.759995
10465  2022-06-15  134.289993  137.339996  132.160004  135.429993  135.429993
10466  2022-06-16  132.080002  132.389999  129.039993  130.059998  130.059998
10467  2022-06-17  130.070007  133.080002  129.809998  131.559998  131.559998

          Volume
0      469033600
1      175884800
2      105728000
3       86441600
4       73449600
...          ...
10463  122207100
10464   84784300
10465   91533000
10466  108123900
10467  134118500

[10468 rows x 7 columns]>
```

```
[ ] df.head()# for some stating data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600

```
df.tail() #for some last data
```

	Date	Open	High	Low	Close	Adj Close	Volume
10463	2022-06-13	132.869995	135.199997	131.440002	131.880005	131.880005	122207100
10464	2022-06-14	133.130005	133.889999	131.479996	132.759995	132.759995	84784300
10465	2022-06-15	134.289993	137.339996	132.160004	135.429993	135.429993	91533000
10466	2022-06-16	132.080002	132.389999	129.039993	130.059998	130.059998	108123900
10467	2022-06-17	130.070007	133.080002	129.809998	131.559998	131.559998	134118500

```
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

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

	Open	High	Low	Close	Adj Close	Volume
count	10468.000000	10468.000000	10468.000000	10468.000000	10468.000000	1.046800e+04
mean	14.757987	14.921491	14.594484	14.763533	14.130431	3.308489e+08
std	31.914174	32.289158	31.543959	31.929489	31.637275	3.388418e+08
min	0.049665	0.049665	0.049107	0.049107	0.038329	0.000000e+00
25%	0.283482	0.289286	0.276786	0.283482	0.235462	1.237768e+08
50%	0.474107	0.482768	0.465960	0.475446	0.392373	2.181592e+08
75%	14.953303	15.057143	14.692589	14.901964	12.835269	4.105794e+08
max	182.630005	182.940002	179.119995	182.009995	181.511703	7.421641e+09

```
# Sklearn accepts x (input data) in 2 dimensional numpy array ONLY
# x always needs to be in 2 dimensional Numpy array ONLY
# y can be 1 dimensional numpy array

x = df[['Date']].values
```

```
[ ] x.ndim
```

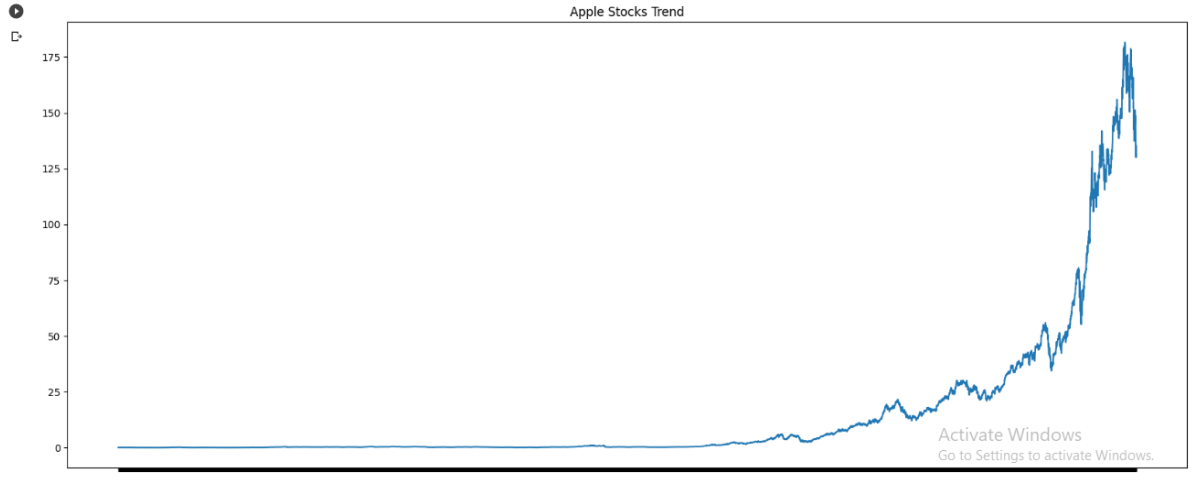
```
2
```

```
[ ] y = df['Adj Close'].values
```

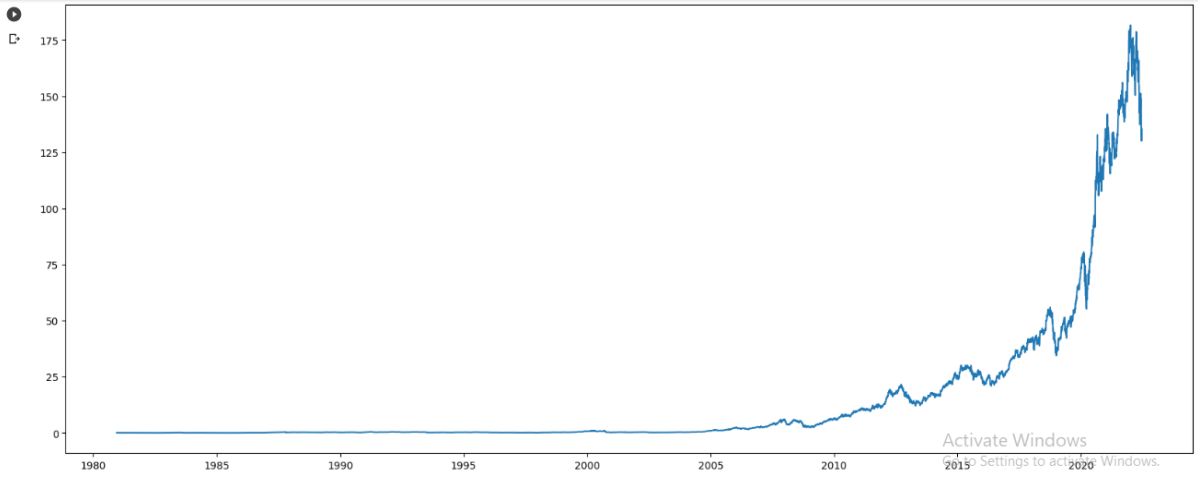
```
[ ] y.ndim
```

```
1
```

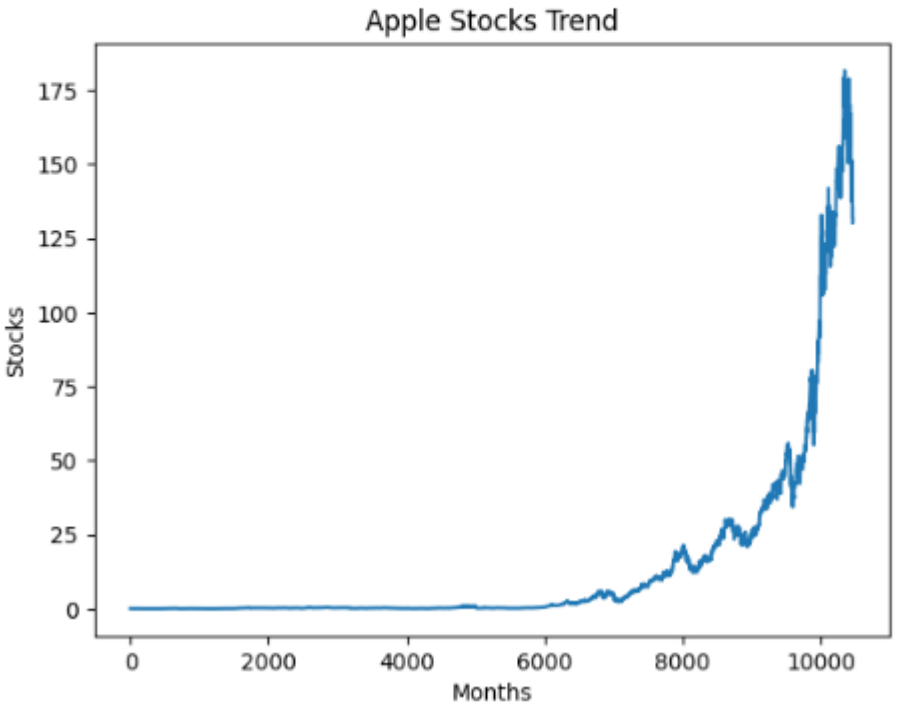
```
import matplotlib.pyplot as plt #for plot the graph
plt.figure(figsize=[20,8])
plt.plot(df['Date'],df['Adj Close'])
plt.title('Apple Stocks Trend')
plt.show() #original data points
```



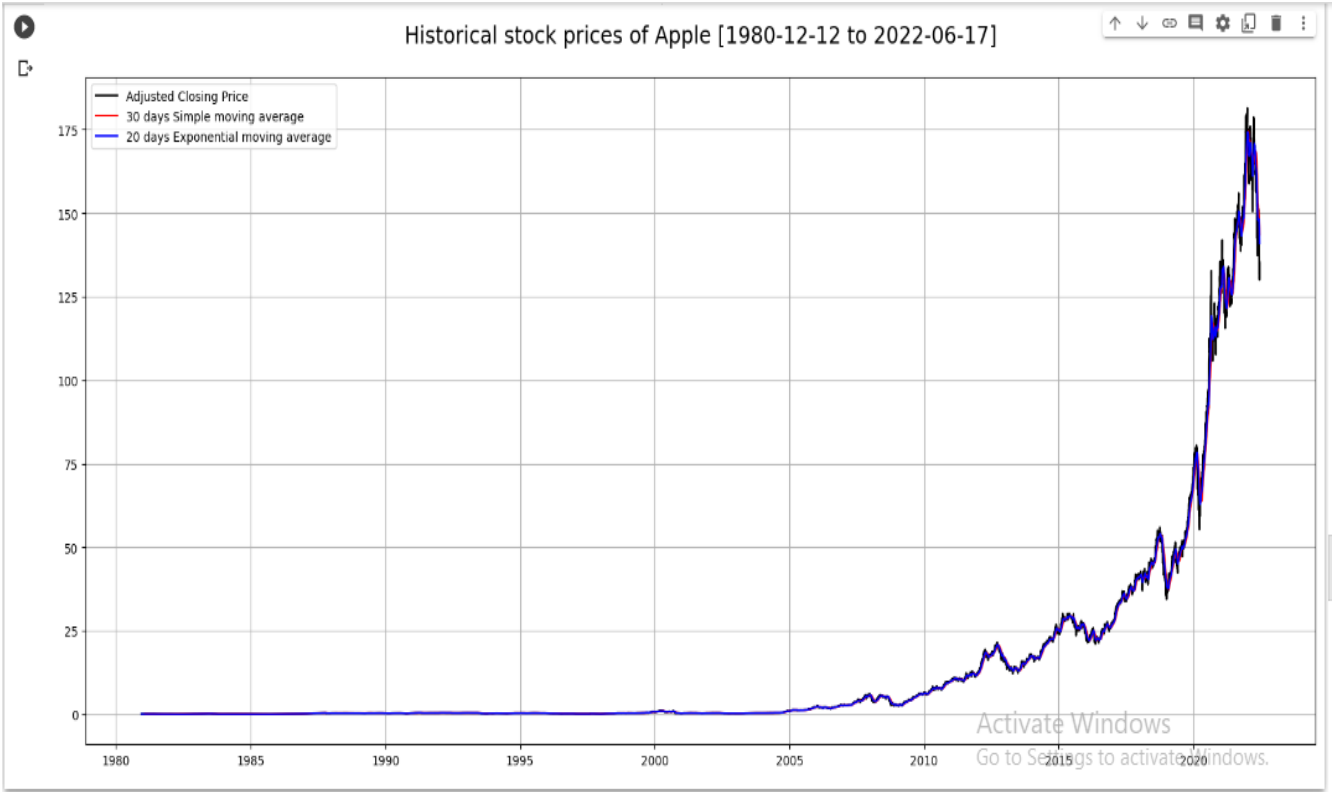
```
plt.figure(figsize=[20,8])
date=df['Date']
ad_cls=df['Adj Close']
plt.plot(date,ad_cls)
plt.show()
```




```
plt.ylabel('Stocks');
plt.xlabel('Months');
plt.plot(df['Adj Close'])
plt.title('Apple Stocks Trend')
plt.show()
```



```python
#converting in datetime object
start_date = pd.to_datetime('1980-12-12')
end_date = pd.to_datetime('2022-06-17')
df['Date'] = pd.to_datetime(df['Date'])
#convert the 'Date' column of the DataFrame into datetime objects .
new_df = (df['Date']>= start_date) & (df['Date']<= end_date)
#create new data frame
df1 = df.loc[new_df]
stock_data = df1.set_index('Date')
close_px = stock_data['Adj Close']
#cal of 30 days Simple Moving Average (SMA) and 20 days Exponential Moving Average (EMA)
stock_data['SMA_30_days'] = stock_data.iloc[:,4].rolling(window=30).mean()
stock_data['EMA_20_days'] = stock_data.iloc[:,4].ewm(span=20,adjust=False).mean()
#sets the size of the figure
plt.figure(figsize=[15,10])
#adds a grid
plt.grid(True)
#set the title
plt.title('Historical stock prices of Apple [1980-12-12 to 2022-06-17]\n',fontsize=18, color='black')
#plot the adjusted closing price, 30-day SMA, and 20-day EMA
plt.plot(stock_data['Adj Close'],label='Adjusted Closing Price', color='black')
plt.plot(stock_data['SMA_30_days'],label='30 days Simple moving average', color='red')
plt.plot(stock_data['EMA_20_days'],label='20 days Exponential moving average', color='blue')
#add the legend to the upper corner
plt.legend(loc=2)
plt.show()
```



Simple Linear Regression

```
[ ] from sklearn.linear_model import LinearRegression #import the algorithm
```

```
[ ] X = df[['Volume']]
    y = df['Close']
```

```
[ ] model = LinearRegression()
    #training the machine / fitting the model
    model.fit(X,y)
```

```
LinearRegression()
```

```
[ ] # testing part of the model (test data - input)
    y_pred = model.predict(X)
    y_pred  # predicted output
```

```
array([12.20600857, 17.63160955, 18.93007232, ..., 19.19279353,
       18.88572898, 18.40462066])
```

```
[ ] y #original value
```

```
0          0.128348
1          0.121652
2          0.112723
3          0.115513
4          0.118862
            ...
10463    131.880005
10464    132.759995
10465    135.429993
10466    130.059998
10467    131.559998
Name: Close, Length: 10468, dtype: float64
```

```
# ACCURACY
#create a new dataframe for the actual vs predicted output

df_new = pd.DataFrame({'Actual Output':y,'Predicted Output':y_pred})
df_new
```

	Actual Output	Predicted Output
0	0.128348	12.206009
1	0.121652	17.631610
2	0.112723	18.930072
3	0.115513	19.287025
4	0.118862	19.527481
...	...	...
10463	131.880005	18.625077
10464	132.759995	19.317699
10465	135.429993	19.192794
10466	130.059998	18.885729
10467	131.559998	18.404621

10468 rows × 2 columns

```
[ ] # (r2 score for regression)       # coefficient of determination /regression score
    from sklearn.metrics import r2_score
    r2_score(y,y_pred)
```

```
0.03857710118534097
```

```
[ ] df[15:20]
```

	Date	Open	High	Low	Close	Adj Close	Volume
15	1981-01-06	0.144531	0.144531	0.143973	0.143973	0.112374	45158400
16	1981-01-07	0.138393	0.138393	0.137835	0.137835	0.107583	55686400
17	1981-01-08	0.135603	0.135603	0.135045	0.135045	0.105406	39827200
18	1981-01-09	0.142299	0.142857	0.142299	0.142299	0.111067	21504000
19	1981-01-12	0.142299	0.142299	0.141183	0.141183	0.110196	23699200

```
[ ] model.predict([[23699200]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
array([20.44826219])
```

```
[ ] # y = m*x+c
    model.coef_           # m value or slope value
```

```
array([-1.85080102e-08])
```

```
model.intercept_       # c value or y-intercept value
```



```
20.886887228304214
```

```
[ ] -1.85080102e-08*23699200+20.886887228304214
```

```
20.448262192972376
```

```
model.predict([[21504000]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
array([20.48889098])
```

```
[ ] -1.85080102e-0*21504000+20.886887228304214
```

```
-39799604.24719277
```

```
[ ] X1 = df[['Open', 'High', 'Low','Close','Adj Close', 'Volume']]
    y1 = df['Close']
```

```
[ ] # Create the linear regression model and fit it to the data
    model = LinearRegression()
    model.fit(X1, y1)
```

```
LinearRegression
LinearRegression()
```

```
[ ] y_pred = model.predict(X1)
    y_pred  # predicted output
```

```
array([1.28348000e-01, 1.21652000e-01, 1.12723000e-01, ...,
       1.35429993e+02, 1.30059998e+02, 1.31559998e+02])
```

```
[ ] y1
```

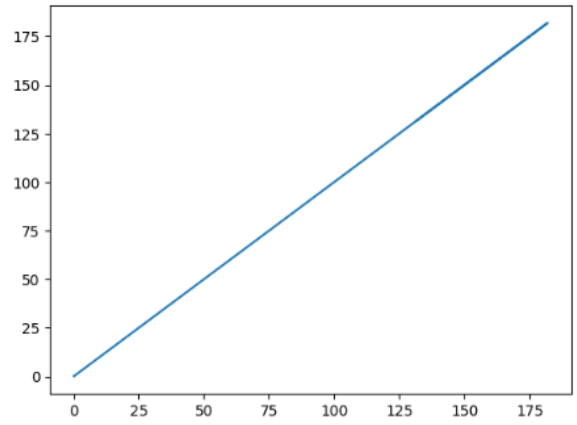
```
0          0.128348
1          0.121652
2          0.112723
3          0.115513
4          0.118862
             ...
10463    131.880005
10464    132.759995
10465    135.429993
10466    130.059998
10467    131.559998
Name: Close, Length: 10468, dtype: float64
```

```
from sklearn.metrics import r2_score
r2_score(y,y_pred)
```

```
1.0
```

```
plt.plot(y,y_pred)
```

```
[<matplotlib.lines.Line2D at 0x7fb54ce29820>]
```




```
[ ] print(model.coef_)
```

```
[-1.38671104e-13  1.39435991e-13 -3.45040739e-15  1.00000000e+00
 -9.54512970e-16  1.97993951e-20]
```