In [ ]:

```
#This is specific functions I have created for training different  Ai models by fetching stoc
# Import specific functions from finance_analytics.py
from finance_analytics import split_dataset, data_preprocess, best_model_selector , fetch_sto
```

In [ ]:

```
#Importing the specific Libraries

import pandas as pd
import numpy as np
import yfinance as yf
import sklearn

#Libraries for the higher models
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer


import yfinance as yf
import pandas as pd
```

In [ ]:

```
#Fetching the stock data using the function fetch_stock_data

# fetch_stock_data(ticker symbol,start date,end date)

fetch_stock_data("AAPL","2020-01-01","2023-01-01")
```

```
[*********************100%%**********************]  1 of 1 completed
```

Out[ ]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-01-02	74.059998	75.150002	73.797501	75.087502	72.960472	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.251122	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.826866	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.484329	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.650345	132079200
...	...	...	...	...	...	...
2022-12-23	130.919998	132.419998	129.639999	131.860001	130.782578	63814900
2022-12-27	131.380005	131.410004	128.720001	130.029999	128.967529	69007800
2022-12-28	129.669998	131.029999	125.870003	126.040001	125.010124	85438400
2022-12-29	127.989998	130.479996	127.730003	129.610001	128.550964	75703700
2022-12-30	128.410004	129.949997	127.430000	129.929993	128.868332	77034200

756 rows × 6 columns

In [ ]:

```
#Storing the values of the fetched_stock_data in a variable
df = fetch_stock_data("AAPL","2020-01-01","2023-01-01")
```

```
[*********************100%%***********************]  1 of 1 completed
```

In [ ]:

```
#Viewing the info about the created dataframe

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 756 entries, 2020-01-02 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       756 non-null    float64
 1   High       756 non-null    float64
 2   Low        756 non-null    float64
 3   Close      756 non-null    float64
 4   Adj Close  756 non-null    float64
 5   Volume     756 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 41.3 KB
```

In [ ]:

```
#Using the split_data function to split the stock data , split_data(dataset,target column, te

# Call the function to split the dataset
split_data  = split_dataset(df,"Close",0.2)
```

In [ ]:

```
#Storing the split data into various variables

X_train, X_test, y_train, y_test = split_data
```

In [ ]:

```
#Viewing the split datasets created
#I have just show the table created for one split dataset you can try viewing other datasets.

X_train.head(6)
```

Out[ ]:

Date	Open	High	Low	Adj Close	Volume
2020-11-03	109.660004	111.489998	108.730003	108.051842	107624400
2020-04-13	67.077499	68.425003	66.457497	66.534904	131022800
2021-04-23	132.160004	135.119995	132.160004	131.838913	78657500
2020-07-10	95.334999	95.980003	94.705002	93.676926	90257200
2020-03-06	70.500000	72.705002	70.307503	70.377251	226176800
2020-04-14	70.000000	72.062500	69.512497	69.895142	194994800

In [ ]:

```
X_train.shape
```

Out[ ]:

```
(604, 5)
```

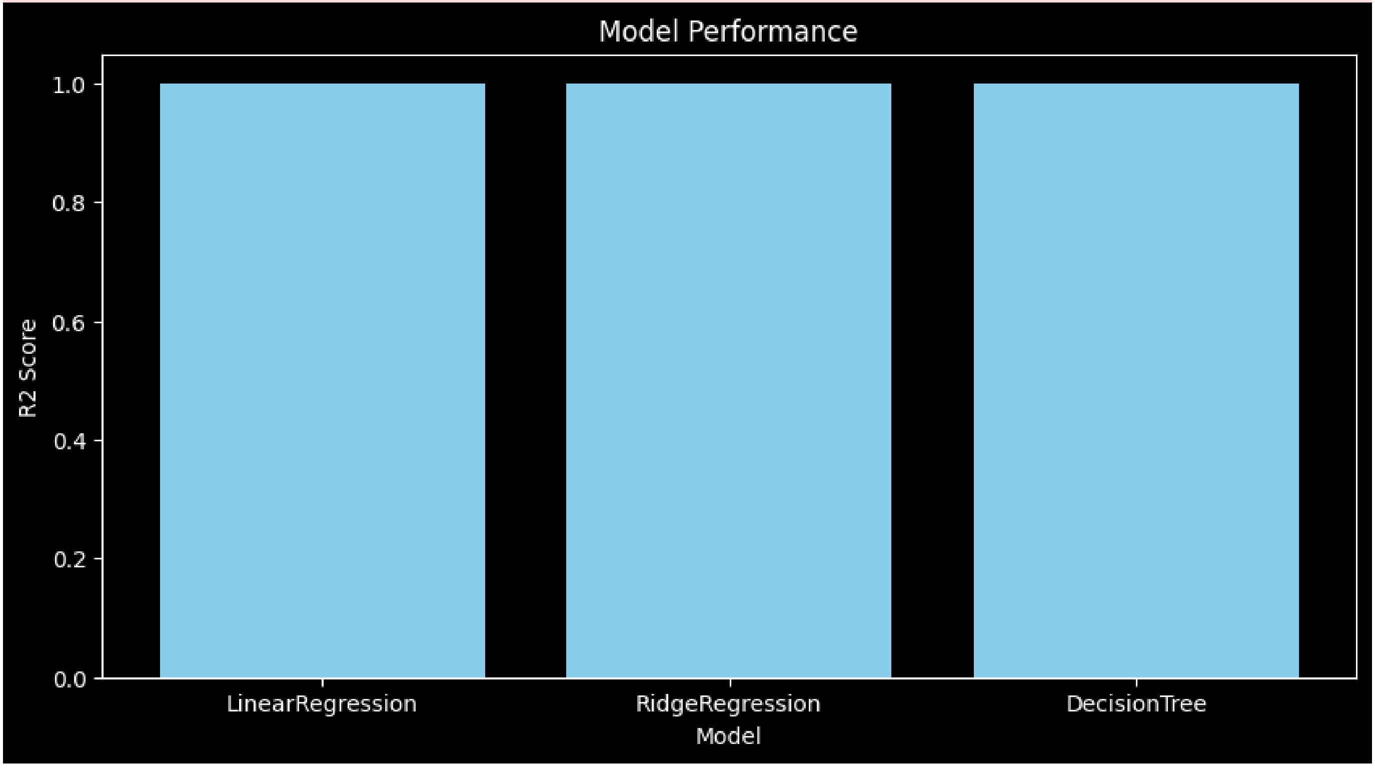
In [ ]:

```
#Using the best model selector function

#This function applies all the models mentioned in the testing_models along with the selected
# its score and comparison graph.
```

```
testing_models = ("LinearRegression","RidgeRegression","DecisionTree")
evaluation_method = "r2"

best_model_selector(X_train, X_test, y_train, y_test, testing_models, evaluation_method)
```

```
The best model is: LinearRegression with a r2 score of 0.9999
```

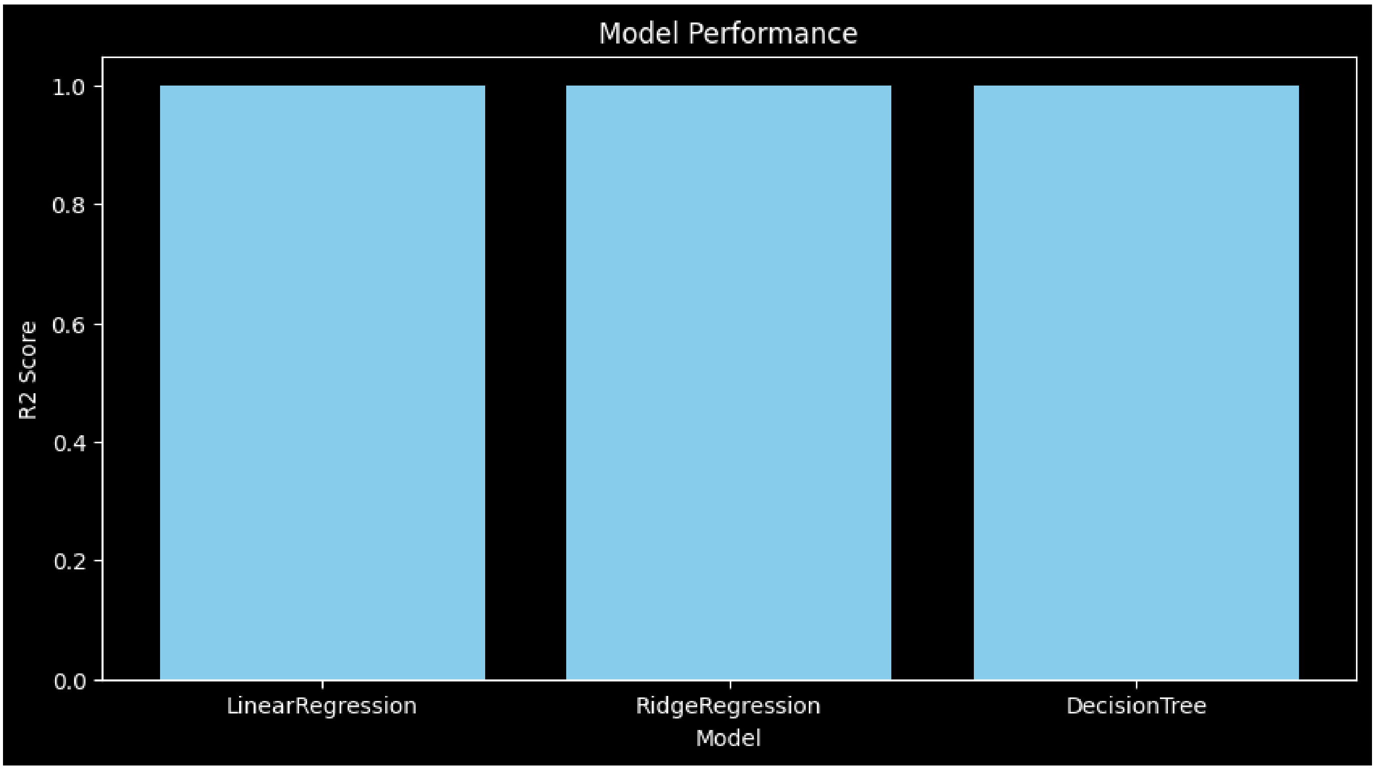
```
C:\Users\Mudassir\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\Lo
calCache\local-packages\Python312\site-packages\sklearn\linear_model\_ridge.py:204: LinAlgWarn
ing: Ill-conditioned matrix (rcond=8.17113e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```



```
Out[ ]:
▾ LinearRegression
LinearRegression()
```

```
In [ ]: final_model = best_model_selector(X_train, X_test, y_train, y_test, testing_models, evaluatio
```

```
C:\Users\Mudassir\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\Lo
calCache\local-packages\Python312\site-packages\sklearn\linear_model\_ridge.py:204: LinAlgWarn
ing: Ill-conditioned matrix (rcond=8.17113e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
The best model is: LinearRegression with a r2 score of 0.9999
```



In [ ]:

```
#predicting the values using the final best selected model

final_model.predict(X_test)
```

```
Out[ ]: array([150.62921641,  79.38633653, 154.81449057, 156.85764245,
               140.58341379,  70.85779617,  58.26390819,  85.94097386,
               134.25346033, 176.13099487,  68.98803461, 123.55060614,
               124.29054617, 163.27828542, 161.22744738, 133.18761646,
               138.2819867 , 143.58465601, 148.46640048, 168.89206585,
               144.11638711, 154.35494059, 121.8623523 , 151.26477447,
               160.24579807, 142.86447051, 141.08756157, 133.8486252 ,
               145.74352573, 113.02690137, 125.85855965, 119.0838786 ,
               114.94933923,  90.17238689, 123.13506089, 122.84513513,
               147.96299223, 135.54338773, 155.87429199, 149.11971997,
               179.49636637, 172.08715914,  65.27161221, 130.5882627 ,
               145.04626905, 143.40653256, 150.13773404, 120.17164127,
               114.84496311, 130.79141557,  79.98966321, 124.84887412,
                69.0658004 , 169.62266118, 109.42964825, 161.29591622,
               108.69511784, 165.43914663, 124.94308736,  78.61115191,
               128.27379021,  72.1006044 ,  79.78566945, 162.6317581 ,
               148.69560026, 165.28743381, 124.49539793, 132.24618505,
               136.62110613, 156.24173444,  65.83078971, 155.9636621 ,
               165.15073648, 123.86763577, 143.2907869 , 131.43962529,
               137.86752195, 136.31307519, 126.08409476,  69.1883074 ,
               150.8761711 ,  71.80883564, 118.88741232, 145.99488749,
               131.67983858,  61.52674433, 173.60397033, 162.81546042,
                69.59311375, 126.78661322, 112.86915073,  79.80113483,
               159.43373485, 148.41607005, 150.90720665,  79.01607478,
               131.97690248,  80.13326932, 145.59227604, 132.5933628 ,
               121.92637651, 123.42790839, 166.03467543, 150.98310977,
               115.19446799,  89.66017088,  75.31456387, 126.82566751,
                74.77263187, 175.40571947, 171.0162686 ,  81.10460237,
               136.96180649, 138.93110655, 165.12209131, 108.49292194,
               178.23271993, 153.93836185, 145.67887639, 161.97583099,
               109.39338901, 134.9258145 , 146.57194464, 120.08528513,
               142.85339652, 132.31832432,  73.31148202, 146.40002728,
                70.92810274, 126.85277005, 167.24344224, 163.80851987,
               110.96613449,  96.78414097, 163.94226956,  56.63081676,
               178.08239315, 126.84560487, 137.67932613, 143.80980385,
               121.42303775, 131.34425471,  64.05279518, 133.37255896,
               179.12537834, 135.72908799, 132.69035428, 114.4799159 ,
               160.47154325, 130.3692893 ,  78.15811941, 160.95729703])

In [ ]:
```