```
# Import the Dependencies
import pandas as pd
import numpy as np
from sklearn.tree  import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import  train_test_split
import matplotlib.pyplot as plt
plt.style.use('bmh')

# Load the Data
from google.colab import files
upload = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving NFLX_Stock_Price.csv to NFLX_Stock_Price (1).csv
```

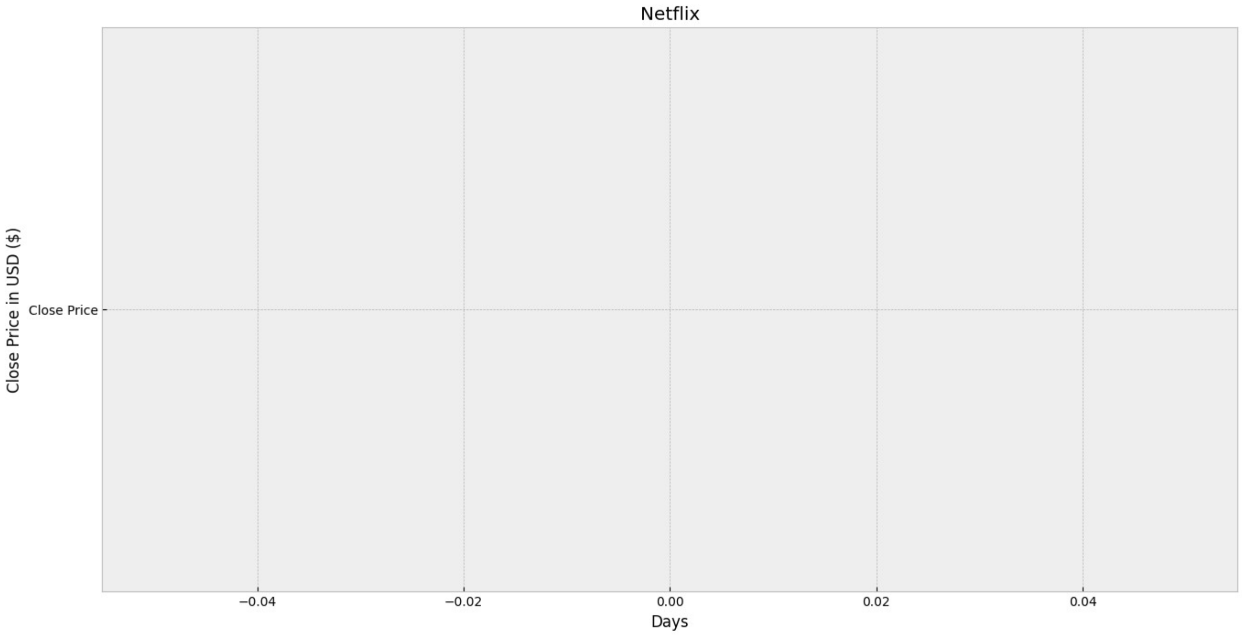
```
# Store data the data into a dataframe
df = pd.read_csv("NFLX_Stock_Price.csv")

# Print the first 6 rows

df.head(6)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1256,\n  \"fields\":
[\n    {\n      \"column\": \"Date\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 1256,\n
\"samples\": [\n          \"28-07-2020\",\n          \"01-04-2019\",\n
\"17-01-2019\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Open\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 116.88975903924417,\n        \"min\": 163.960007,\n
\"max\": 692.349976,\n        \"num_unique_values\": 1205,\n
\"samples\": [\n          369.26001,\n          242.669998,\n
377.179993\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"High\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 117.56414162846733,\n        \"min\": 172.059998,\n
\"max\": 700.98999,\n        \"num_unique_values\": 1228,\n
\"samples\": [\n          592.97998,\n          490.059998,\n
291.450012\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Low\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 115.78647189186238,\n        \"min\": 162.710007,\n
\"max\": 686.090027,\n        \"num_unique_values\": 1227,\n
\"samples\": [\n          360.679993,\n          325.529999,\n
305.75\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Close\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 116.57838026718646,\n        \"min\": 166.369995,\n
\"max\": 691.690002,\n        \"num_unique_values\": 1237,\n
\"samples\": [\n          510.299988,\n          325.209991,\n
204.009995\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Adj
Close\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 116.57838026718646,\n        \"min\": 166.369995,\n
\"max\": 691.690002,\n        \"num_unique_values\": 1237,\n
\"samples\": [\n          510.299988,\n          325.209991,\n
204.009995\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Volume\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 6474654,\n        \"min\": 1144000,\n        \"max\":
133387500,\n        \"num_unique_values\": 1243,\n        \"samples\":
[\n          6949000,\n          11124700,\n          2777200\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
# Get the number of  training days
df.shape
```

```
(1256, 7)
```

```
# Visualizing the close price data
plt.figure(figsize=(16,8))
plt.title(('Netflix'))
plt.xlabel('Days')
plt.ylabel('Close Price in USD ($)')
plt.plot('Close Price')
plt.show()
```



```
# Get the close Price
df = df[['Close']]
df.head(4)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1256,\n  \"fields\":
[\n    {\n      \"column\": \"Close\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 116.57838026718646,\n
\"min\": 166.369995,\n        \"max\": 691.690002,\n
\"num_unique_values\": 1237,\n        \"samples\": [\n
510.299988,\n          325.209991,\n          204.009995\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
# Create a varuiable to predict the 'X"  days out into the future
future_days = 25
# Create a new column (targets)
df['Prediction'] = df[['Close']].shift(-future_days)
df.head(4)
df.tail(4)
```

```
<ipython-input-20-e3b58f4b6b26>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['Prediction'] = df[['Close']].shift(-future_days)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 4,\n  \"fields\": [\n
{\n      \"column\": \"Close\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6.217191897479511,\n
\"min\": 397.869995,\n        \"max\": 411.25,\n
\"num_unique_values\": 4,\n        \"samples\": [\n
403.540009,\n          410.079987,\n          411.25\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Prediction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
null,\n        \"min\": null,\n        \"max\": null,\n
\"num_unique_values\": 0,\n        \"samples\": [],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```
# Create the feature dataset (X) and convert it to a numpy array and
remove the last 'X'
X = np.array(df.drop(['Prediction'], axis = 1))[:-future_days]
print(X)
```

```
[[317.380005]
 [309.100006]
 [315.440002]
 ...
 [384.149994]
 [379.809998]
 [384.799988]]
```

```
# Create the target dataset  (Y) convert it into a np.array and get
all the target values
Y = np.array(df['Prediction'][:-future_days])
print(Y)
```

```
[269.700012 265.320007 274.880005 ... 403.540009 397.869995
410.079987]
```

```
# Split the data into 75% training data and 25% testing data
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size
=0.25)

# Create the nodels
# Create the Decision tree  regression model
tree = DecisionTreeRegressor().fit(X_train, Y_train)
#Create the linear regression model
lr = LinearRegression().fit(X_train, Y_train)

# Get the last X rows of the feature data set
X_future = df[['Close']].iloc[:-future_days]
X_future = X_future.tail(future_days)
X_future = np.array(X_future)
X_future
X_future
```

```
array([[408.290009],
       [413.170013],
       [427.549988],
       [406.929993],
       [416.029999],
       [418.059998],
       [429.98999 ],
       [434.670013],
       [433.679993],
       [439.880005],
       [448.679993],
       [445.76001 ],
       [443.140015],
       [442.799988],
       [445.359985],
       [434.690002],
       [412.23999 ],
       [400.48999 ],
       [396.940002],
       [394.399994],
       [396.200012],
       [386.299988],
       [384.149994],
       [379.809998],
       [384.799988]])
```

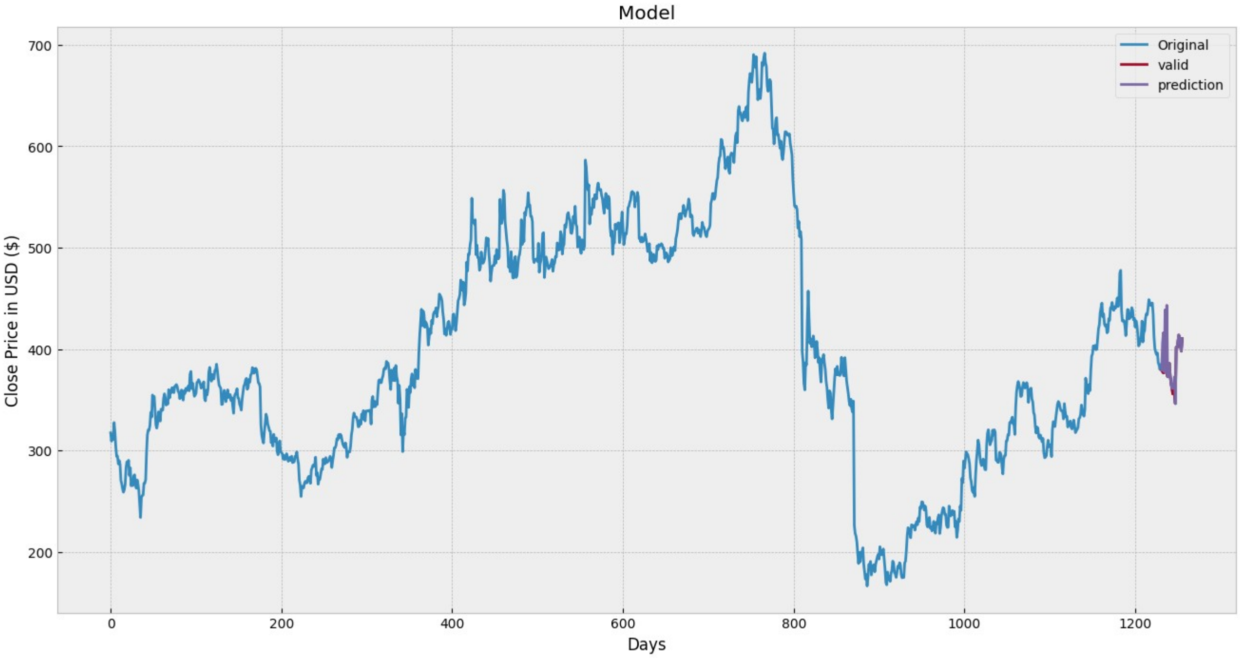
```
# Show the modela tree prediction
tree_prediction = tree.predict(X_future)
print(tree_prediction)
print()
# Show the model linear regression model
lr_prediction =lr.predict(X_future)
print(lr_prediction)
print()
```

```
[379.25      405.6349945 416.029999  377.600006  438.619995  376.75
 443.140015  372.589996  381.51001   385.950012  373.320007
365.929993
 361.200012  361.200012  365.929993  372.589996  346.190002
401.769989
 400.959991  406.839996  413.730011  411.25      403.540009
397.869995
 410.079987 ]

[407.93212654 412.21033372 424.81698574 406.73982625 414.71762934
 416.49729106 426.95608933 431.05897695 430.19104515 435.62647837
 443.34126136 440.78136751 438.48446746 438.18637222 440.43067316
 431.07650093 411.39500015 401.09399746 397.98179014 395.75501318
 397.33305491 388.65389117 386.76903211 382.96423973 387.33886955]
```

```
# Visualize the data
prediction = tree_prediction

valid = df[X.shape[0]:]
valid['Predictions'] = prediction
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Days')
plt.ylabel('Close Price in USD ($)')
plt.plot(df['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Original','valid', 'prediction'])
plt.show()
```

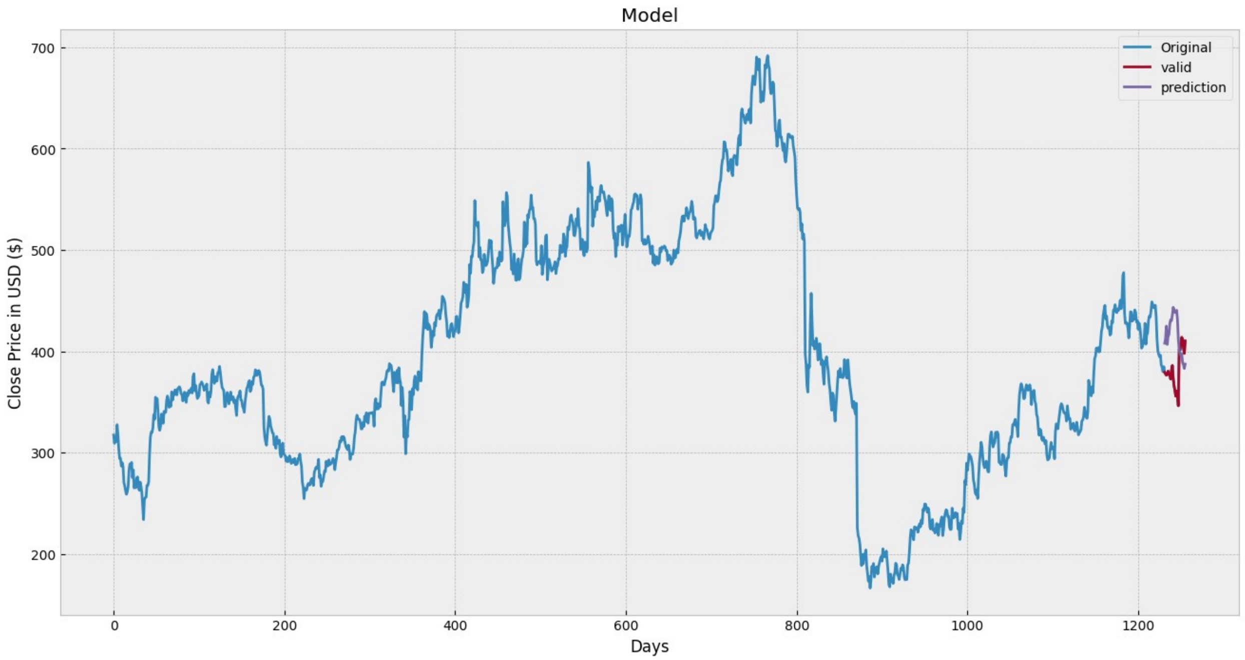
```
<ipython-input-48-34fa502df842>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
```

```
returning-a-view-versus-a-copy
  valid['Predictions'] = prediction
```



```
# visualize the data
prediction = lr_prediction

valid = df[X.shape[0]:]
valid['Predictions'] = prediction
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Days')
plt.ylabel('Close Price in USD ($)')
plt.plot(df['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Original','valid', 'prediction'])
plt.show()
```

```
<ipython-input-50-94dd616af78c>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  valid['Predictions'] = prediction
```


Model
Original
valid
prediction
Close Price in USD ($)
700
600
500
400
300
200
0
200
400
600
800
1000
1200
Days