Projects Report

INT 247

Topic:

Share Market Prediction using Machine Learning

Name: Rajnish Kumar

Reg No.: 11905327

Section: KM044

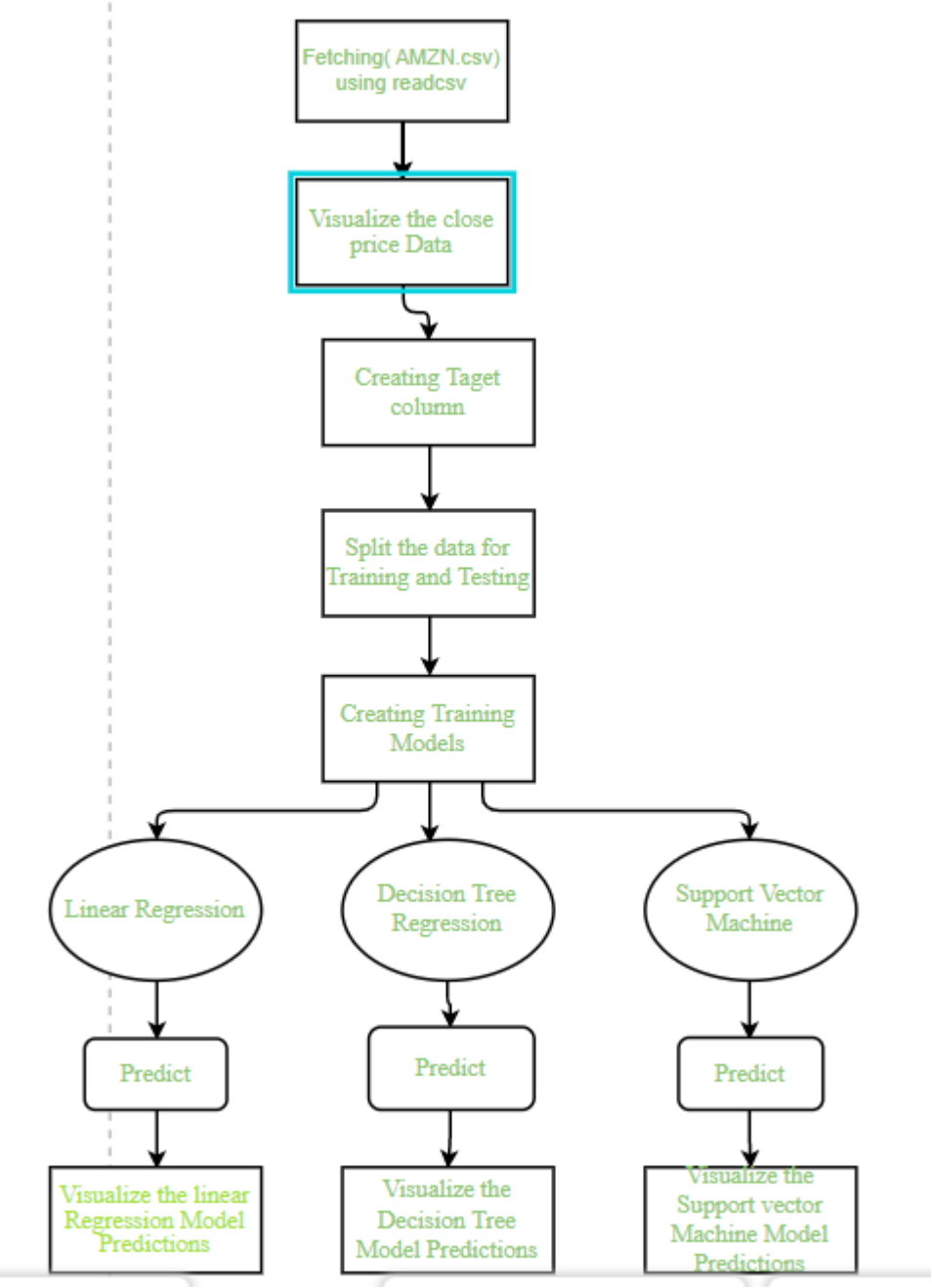
Introduction

Machine Learning (ML) has a wide range of industrial applications that are likely to increase in the coming areas. Regression algorithms in Machine Learning are an important concept with a lot of use cases.

The future values are predicted with the help of regression algorithms in Machine Learning. The input data/historical data is used to predict a wide range of future values using regression. Label in ML is defined as the target variable (to be predicted) and regression helps in defining the relationship between label and data points. Regression is a type of supervised learning in ML that helps in mapping a predictive relationship between labels and data points. The top types of regression algorithms in ML are Linear Regression, Decision Tree and Support vector machine etc.

In this project also we have to find or predict the future value of stock price of different company using different algorithm like as linear regression, decision tree and support vector algorithm so that we can find that which algorithm is more effective and which algorithm is less effective.

FlowChart



WorkDone:

File Edit View Insert Cell Kernel Help Not Trusted | Python 3 (ipykernel)

Stock Price Prediction Using Machine Learning

- Decision tree Regression
- Linear Regression
- Support Vector Machines

In [96]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
amazon=pd.read_csv("AMZN.csv")
print(amazon.head())
```

```
         Date         Open         High          Low        Close  \
0  2021-04-13  3400.850098  3432.000000  3395.629883  3400.000000
1  2021-04-14  3404.040039  3404.129883  3326.000000  3333.000000
2  2021-04-15  3371.000000  3397.000000  3352.000000  3379.090088
3  2021-04-16  3380.000000  3406.800049  3355.590088  3399.439941
4  2021-04-19  3390.330078  3435.929932  3360.159912  3372.010010

     Adj Close   Volume
0  3400.000000  3315800
1  3333.000000  3145200
2  3379.090088  3233600
3  3399.439941  3186000
4  3372.010010  2725400
```

To Visualize the close price Data

In [97]:
```
sns.set()
plt.figure(figsize=(15,5))
plt.title("Amazon Stock Price")
plt.xlabel("Days")
plt.ylabel("Close Price USD")
plt.plot(amazon["Close"])
plt.show()
```

To get the Close Price

In [98]:
```
amazon=amazon[['Close']]
print(amazon.head())
```

```
        Close
0  3400.000000
1  3333.000000
2  3379.090088
3  3399.439941
4  3372.010010
```

In [99]:
```
FutureDays=30 #Future prediction days
```

Creating Taget column (Prediction)

In [100]:
```
amazon["Prediction"]=amazon['Close'].shift(-FutureDays)
```

In [101]:
```
print(amazon.head())
print(amazon.tail())
```

```
         Close   Prediction
0  3400.000000  3259.050049
1  3333.000000  3265.159912
2  3379.090088  3230.110107
3  3399.439941  3223.070068
4  3372.010010  3218.649902
           Close  Prediction
249  3175.120117         NaN
250  3155.689941         NaN
251  3089.209961         NaN
252  3022.439941         NaN
253  3015.750000         NaN
```

In [102]:
```
import numpy as np
x=np.array(amazon.drop(["Prediction"],1))[:-FutureDays]
print(x)
```

```
 [3151.939941]
 [3161.469971]
 [3222.899902]
 [3270.389893]
 [3232.280029]
 [3231.800049]
 [3247.679932]
 [3203.080078]
 [3244.98999 ]
 [3259.050049]
 [3265.159912]
 [3230.110107]
 [3223.070068]
 [3218.649902]
 [3233.98999 ]
 [3187.01001 ]
 [3206.219971]
 [3198.01001 ]
 [3264.110107]
 [3281.149902]
 [3349.649902]
 [3346.830078]
 [3383.870117]
 [3383.129883]
```

In [103]:
```
y = np.array(amazon["Prediction"])[:-FutureDays]
print(y)
```

```
[3259.050049 3265.159912 3230.110107 3223.070068 3218.649902 3233.98999
 3187.01001  3206.219971 3198.01001  3264.110107 3281.149902 3349.649902
 3346.830078 3383.870117 3383.129883 3415.25     3489.23999  3486.899902
 3453.959961 3505.439941 3503.820068 3449.080078 3401.459961 3443.889893
 3448.139893 3440.159912 3432.969971 3510.97998  3675.73999  3696.580078
 3731.409912 3719.340088 3718.550049 3677.360107 3681.679932 3631.199951
 3573.629883 3549.590088 3573.189941 3585.199951 3638.030029 3656.639893
 3699.820068 3626.389893 3630.320068 3599.919922 3327.590088 3331.47998
 3366.23999  3354.719971 3375.98999  3344.939941 3341.870117 3320.679932
 3292.110107 3303.5      3293.969971 3298.98999  3241.959961 3201.219971
 3187.75     3199.949951 3265.870117 3305.780029 3299.179932 3316.
 3349.629883 3421.570068 3470.790039 3479.       3463.120117 3478.050049
 3509.290039 3525.5      3484.159912 3469.149902 3457.169922 3450.
 3475.790039 3488.23999  3462.52002  3355.72998  3343.629883 3380.050049
 3416.       3425.52002  3405.800049 3315.959961 3301.120117 3285.040039
 3283.26001  3189.780029 3221.       3262.01001  3302.429932 3288.620117
 3246.300049 3247.330078 3284.280029 3299.860107 3409.02002  3446.73999
 3444.149902 3415.060059 3435.01001  3335.550049 3320.370117 3376.070068
 3392.48999  3446.570068 3372.429932 3318.110107 3312.75     3384.
 3477.       3518.98999  3488.97998  3576.22998  3482.050049 3472.5
 3525.149902 3545.679932 3540.699951 3549.       3696.060059 3676.570068
 3572.570068 3580.040039 3580.409912 3504.560059 3561.570068 3507.070068
 3443.719971 3437.360107 3389.790039 3427.370117 3523.290039 3523.159912
 3483.419922 3444.23999  3391.350098 3381.830078 3466.300049 3377.419922
 3400.350098 3341.580078 3408.340088 3420.73999  3421.370117 3393.389893
 3413.219971 3384.02002  3372.889893 3334.340088 3408.090088 3350.439941
 3287.139893 3265.080078 3251.080078 3229.719971 3307.23999  3304.139893
 3224.280029 3242.76001  3178.350098 3125.97998  3033.350098 2852.860107
 2890.879883 2799.719971 2777.449951 2792.75     2879.560059 2991.469971
 3023.870117 3012.25     2776.909912 3152.790039 3158.709961 3228.27002
 3223.790039 3180.070068 3065.870117 3103.340088 3130.209961 3162.01001
 3093.050049 3052.030029 3003.949951 2896.540039 3027.159912 3075.77002
 3071.26001  3022.840088 3041.050049 2957.969971 2912.820068 2749.060059
 2720.290039 2785.580078 2936.350098 2910.48999  2837.060059 2947.330078
 3062.080078 3144.780029 3225.01001  3229.830078 3297.780029 3268.159912
 3272.98999  3295.469971 3379.810059 3386.300049 3326.02002  3259.949951
 3271.199951 3366.929932 3281.100098 3175.120117 3155.689941 3089.209961
 3022.439941 3015.75    ]
```

In [104]:
```
#Split the data into 90% training and 10% testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.1)
```

Creating Models

In [105]:
```
# Creating the decision tree regressor model
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor().fit(xtrain, ytrain)

# creating the Linear Regression model
from sklearn.linear_model import LinearRegression
linear = LinearRegression().fit(xtrain, ytrain)

# creating the svm model
from sklearn.svm import SVR
svr = SVR(kernel='rbf', C=1e3, gamma=0.1)
svm = svr.fit(xtrain, ytrain)
```

In [106]:
```
xfuture = amazon.drop(["Prediction"], 1)[:-FutureDays]
xfuture = xfuture.tail(FutureDays)
xfuture = np.array(xfuture)
print(xfuture)
```

```
[[3178.350098]
 [3125.97998 ]
 [3033.350098]
 [2852.860107]
 [2890.879883]
 [2799.719971]
 [2777.449951]
 [2792.75    ]
 [2879.560059]
 [2991.469971]
 [3023.870117]
 [3012.25    ]
 [2776.909912]
 [3152.790039]
 [3158.709961]
 [3228.27002 ]
 [3223.790039]
 [3180.070068]
 [3065.870117]
 [3103.340088]
 [3130.209961]
 [3162.01001 ]
 [3093.050049]
 [3052.030029]
 [3003.949951]
 [2896.540039]
 [3027.159912]
 [3075.77002 ]
 [3071.26001 ]
 [3022.840088]]


C:\Users\acer\AppData\Local\Temp/ipykernel_6212/1555300527.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.d
rop except for the argument 'labels' will be keyword-only
  xfuture = amazon.drop(["Prediction"], 1)[:-FutureDays]
```

In [107]:
```
#To see the model tree prediction
treePrediction = tree.predict(xfuture)
print("Decision Tree prediction =",treePrediction)

#To see the model linear regression prediction
linearPrediction = linear.predict(xfuture)
print("Linear regression Prediction =",linearPrediction)

#To see the model Support vector machine
svmPrediction = svr.predict(xfuture)
print("Support vector machine Prediction =",svmPrediction)
```

```
Decision Tree prediction = [3041.050049 3326.02002  2912.820068 2749.060059 2720.290039 2936.350098
 2936.350098 2936.350098 2837.060059 2947.330078 3062.080078 3144.780029
 3225.01001  3229.830078 3297.780029 3268.159912 3272.98999  3295.469971
 3379.810059 3271.199951 3326.02002  3259.949951 3271.199951 3366.929932
 3281.100098 3175.120117 3155.689941 3089.209961 3022.439941 3015.75    ]
Linear regression Prediction = [3284.54930903 3273.2815387  3253.3516192  3214.51802779 3222.69822902
 3203.08458309 3198.29304395 3201.58494866 3220.26269552 3244.34083843
 3251.31193985 3248.81179636 3198.17685106 3279.04989667 3280.32360626
 3295.28990381 3294.32600686 3284.91937172 3260.34851207 3268.4104192
 3274.19164653 3281.03363319 3266.19645061 3257.37072743 3247.02598702
 3223.91604827 3252.01976056 3262.47854037 3261.50818249 3251.09032243]
Support vector machine Prediction = [3041.15003689 3312.80832132 2912.9200575  2749.15972331 2720.38981291
 3310.17820548 3170.7011318  3310.17820543 2837.16013239 2947.43033515
 3061.9800256  3144.87967673 3224.91018897 3236.87744724 3297.87995243
```

```
3268.0599236  3273.08982666 3295.36996057 3379.71003366 3310.17722559
3325.91972562 3260.04955001 3271.30023111 3366.82977602 3281.19966305
3175.22028037 3155.79024315 3089.30996872 3022.53986594 3015.85001762]
```

Visualize Decision Tree Predictions

In [108]:
```
predictions = treePrediction
valid = amazon[x.shape[0]:]
valid["Predictions"] = predictions
plt.figure(figsize=(20, 6))
plt.title("Amazon Stock Price Prediction Model(Decision Tree Regressor Model)")
plt.xlabel("Days")
plt.ylabel("Close Price USD ($)")
plt.plot(amazon["Close"])
plt.plot(valid[["Close", "Predictions"]])
plt.legend(["Original", "Valid", "Predictions"])
plt.show()
```

```
C:\Users\acer\AppData\Local\Temp/ipykernel_6212/2838319956.py:3: 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"] = predictions
```

Visualize the Linear Model Predictions

In [109]:
```
predictions = linearPrediction
valid = amazon[x.shape[0]:]
valid["Predictions"] = predictions
plt.figure(figsize=(20, 6))
plt.title("Amazon's Stock Price Prediction Model(Linear Regression Model)")
plt.xlabel("Days")
plt.ylabel("Close Price USD ($)")
plt.plot(amazon["Close"])
plt.plot(valid[["Close", "Predictions"]])
plt.legend(["Original", "Valid", "Predictions"])
plt.show()
```

```
C:\Users\acer\AppData\Local\Temp/ipykernel_6212/2713820942.py:3: 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"] = predictions
```

Visualize the Support vector Machine Model Predictions

```
In [110]: predictions = svmPrediction
          valid = amazon[x.shape[0]:]
          valid["Predictions"] = predictions
          plt.figure(figsize=(20, 6))
          plt.title("Amazon Stock Price Prediction Model(Support Vector Machine Model)")
          plt.xlabel("Days")
          plt.ylabel("Close Price USD ($)")
          plt.plot(amazon["Close"])
          plt.plot(valid[["Close", "Predictions"]])
          plt.legend(["Original", "Valid", "Predictions"])
          plt.show()
```

```
C:\Users\acer\AppData\Local\Temp/ipykernel_6212/2276254179.py:3: 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"] = predictions
```


```
In [ ]:
```

Conclusion & Future Scope:

➔ After apply all three machine algorithm we get to know that Decision Tree and Support Vector Machine Algorithm is more efficient for this project.

➔ There are many advance Machine learning algorithm is also there to reach maximum possible accuracy. Some of the algorithm are Long short-term Memory algorithm.