Stock Price Prediction Using Machine Learning

Import Libraries and Dataset

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('ignore')
sns.set_style('darkgrid')
```

In [ ]:

```
dataset = pd.read_csv('ICICI_BANK.csv')
```

Information about the dataset

In [ ]:

```
dataset.shape
```

Out[ ]:

```
(5306, 15)
```

In [ ]:

```
dataset.columns
```

Out[ ]:

```
Index(['Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble'],
      dtype='object')
```

In [ ]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5306 entries, 0 to 5305
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Date                5306 non-null   object 
 1   Symbol              5306 non-null   object 
 2   Series              5306 non-null   object 
 3   Prev Close          5306 non-null   float64
 4   Open                5306 non-null   float64
 5   High                5306 non-null   float64
 6   Low                 5306 non-null   float64
 7   Last                5306 non-null   float64
 8   Close               5306 non-null   float64
 9   VWAP                5306 non-null   float64
 10  Volume              5306 non-null   int64  
 11  Turnover            5306 non-null   float64
 12  Trades              2456 non-null   float64
 13  Deliverable Volume  4789 non-null   float64
 14  %Deliverble         4789 non-null   float64
dtypes: float64(11), int64(1), object(3)
memory usage: 621.9+ KB
```

```
In [ ]: dataset.describe()
```

Out[ ]:

	Prev Close	Open	High	Low	Last	Close	VWAP	
count	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5
mean	550.895392	551.558538	560.558556	541.534197	551.050980	550.995524	551.129031	8
std	368.784064	368.890953	374.079697	363.389664	368.705647	368.725374	368.746905	1
min	67.400000	67.000000	70.450000	66.000000	67.000000	67.400000	68.520000	7
25%	267.562500	267.400000	271.912500	263.625000	267.400000	267.612500	267.577500	9
50%	398.075000	399.000000	406.525000	392.450000	398.700000	398.175000	398.235000	3
75%	873.562500	877.000000	888.775000	859.800000	874.600000	873.562500	873.510000	1
max	1794.100000	1767.050000	1798.150000	1760.150000	1793.000000	1794.100000	1783.460000	2

Data Cleaning

```
In [ ]: display(dataset.head().style.hide_index())
```

Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP
2000-01-03	ICICIBANK	EQ	69.200000	74.350000	74.750000	71.400000	74.750000	74.750000	73.200000
2000-01-04	ICICIBANK	EQ	74.750000	73.050000	78.500000	71.000000	73.250000	73.050000	73.380000
2000-01-05	ICICIBANK	EQ	73.050000	70.000000	73.500000	67.500000	70.000000	69.500000	70.850000
2000-01-06	ICICIBANK	EQ	69.500000	71.000000	74.000000	69.550000	69.750000	70.050000	72.040000
2000-01-07	ICICIBANK	EQ	70.050000	69.000000	72.500000	66.000000	67.000000	67.400000	68.720000

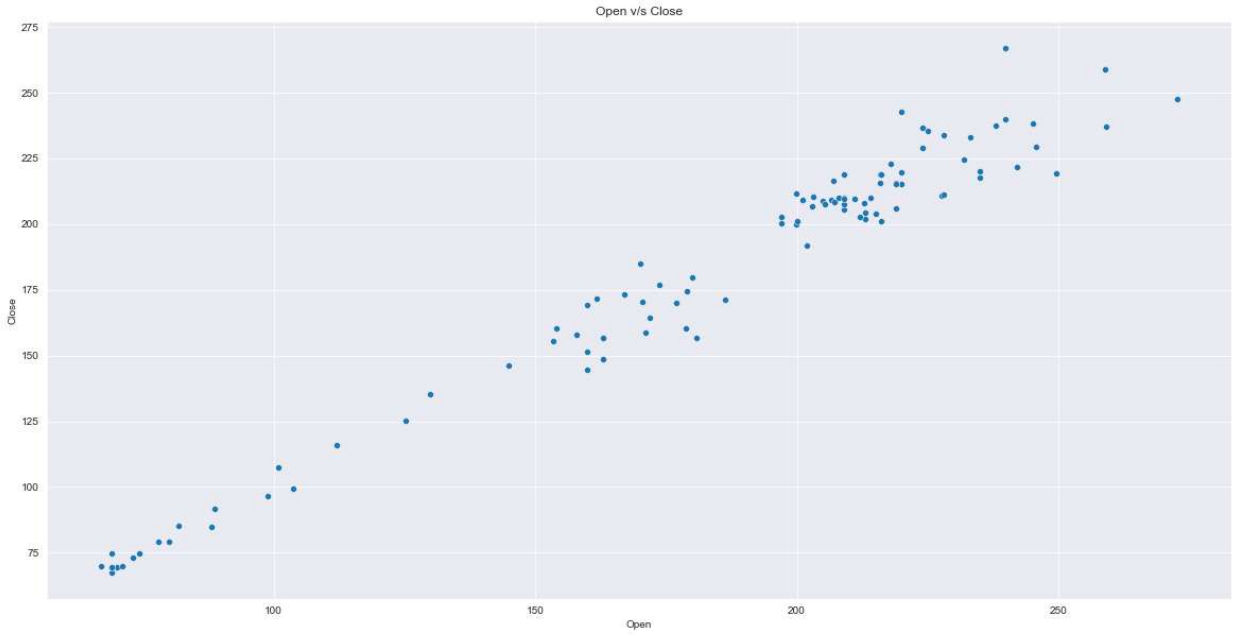
In [ ]:
```
# Delete unnecessary columns

# dataset.drop(["Symbol", "Series", "Prev Close", "High", "Low", "Last", "VWAP", "Volu
# axis = 1, inplace = True)

dataset.drop(dataset.columns.difference(['Date', 'Open', 'Close']), 1, inplace=True)
```

In [ ]:
```
display(dataset.head().style.hide_index())
```

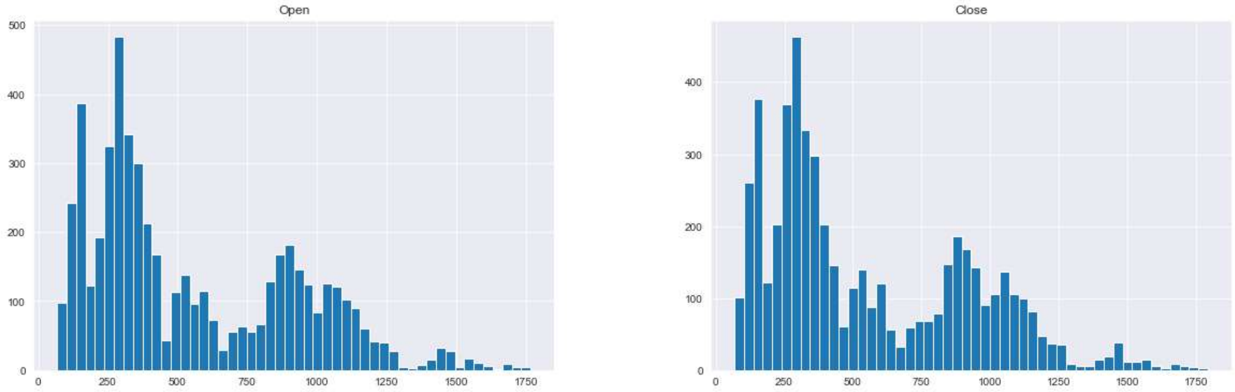
Date	Open	Close
2000-01-03	74.350000	74.750000
2000-01-04	73.050000	73.050000
2000-01-05	70.000000	69.500000
2000-01-06	71.000000	70.050000
2000-01-07	69.000000	67.400000

Data Visualization

In [ ]:
```
fig, ax = plt.subplots(figsize=(20, 10))
plot1 = sns.scatterplot(data=dataset.head(100), x="Open", y="Close", ax=ax)
plot1.set(title='Open v/s Close')
plt.show()
```



In [ ]:
```
dataset.hist(bins=50, figsize=(20, 6))
plt.show()
```



Import Models

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

Build, predict and evaluate models

Simple Linear Regression

In [ ]:
```
X = dataset['Open'].values
y = dataset['Close'].values
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.
```

In [ ]:
```
model1 = LinearRegression()
```

```
build1 = model1.fit(X_train.reshape(-1, 1), y_train)
predict1 = model1.predict(X_test.reshape(-1, 1))
```

In [ ]:
```
print("Co-efficient: ", model1.coef_)
print("\nIntercept: ", model1.intercept_)
```

```
Co-efficient:  [0.99877484]

Intercept:  0.2261374288418665
```

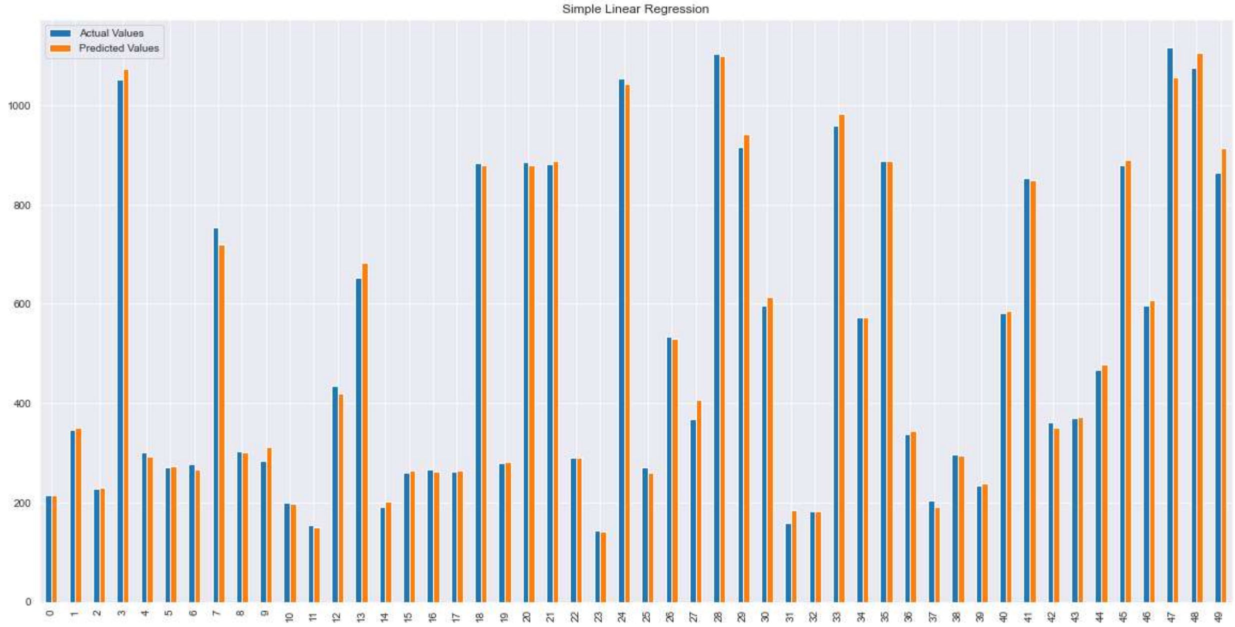
In [ ]:
```
df1 = pd.DataFrame(list(zip(y_test, predict1)), columns=["Actual Values", "Predicted V
```

In [ ]:
```
df1.head().style.hide_index()
```

Out[ ]:

Actual Values	Predicted Values
215.800000	215.761749
346.600000	351.794882
227.750000	229.944351
1051.550000	1072.910319
301.150000	291.768514

In [ ]:
```
df1.head(50).plot(kind="bar", figsize=(20, 10), title='Simple Linear Regression')
plt.show()
```



In [ ]:
```
accuracy1 = r2_score(y_test, predict1)
print("Accuracy of Simple Linear Regression:", accuracy1)
```

```
Accuracy of Simple Linear Regression: 0.9982745129719666
```

Support Vector Regression

In [ ]:
```
model2 = SVR(kernel="rbf", gamma = 0.01, C=100)
build2 = model2.fit(X_train.reshape(-1, 1), y_train)
```

```
predict2 = model2.predict(X_test.reshape(-1, 1))
```

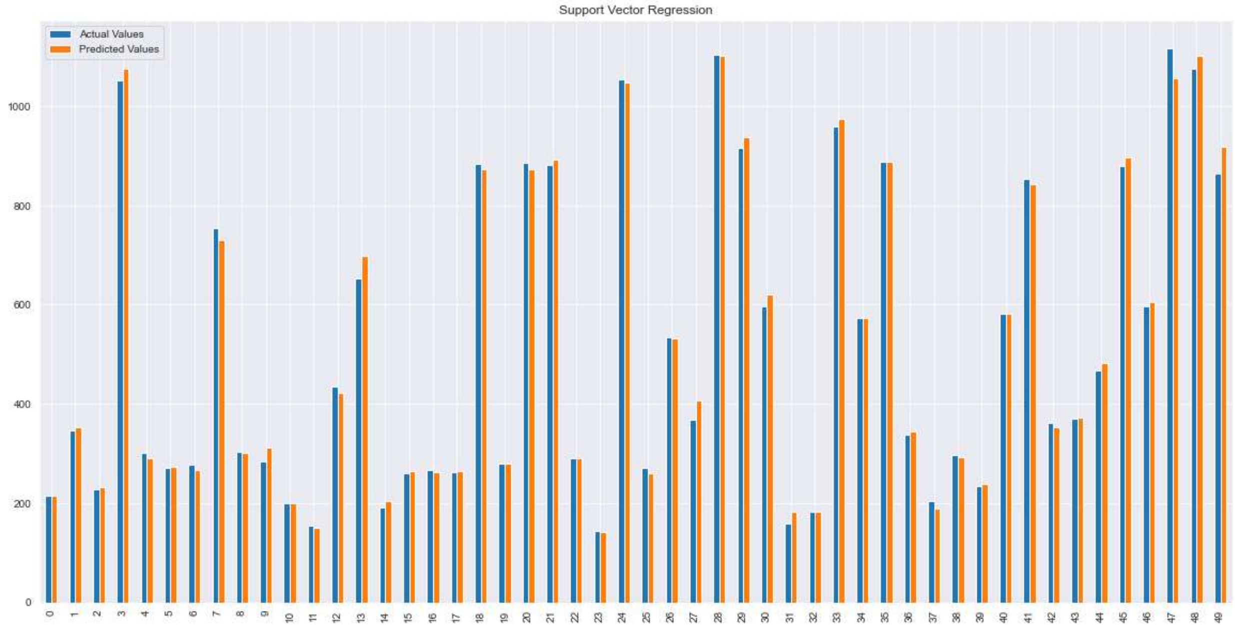
```
In [ ]: df2 = pd.DataFrame(list(zip(y_test, predict2)), columns=["Actual Values", "Predicted V
```

```
In [ ]: df2.head().style.hide_index()
```

Out[ ]:

Actual Values	Predicted Values
215.800000	215.455240
346.600000	352.721438
227.750000	231.609275
1051.550000	1075.539455
301.150000	291.178679

```
In [ ]: df2.head(50).plot(kind="bar", figsize=(20, 10), title='Support Vector Regression')
        plt.show()
```



```
In [ ]: accuracy2 = r2_score(y_test, predict2)
        print("Accuracy of Support Vector Regression:", accuracy2)
```

```
Accuracy of Support Vector Regression: 0.9782539108629036
```

Decision Tree Regression

```
In [ ]: model3 = DecisionTreeRegressor()
        build3 = model3.fit(X_train.reshape(-1, 1), y_train)
        predict3 = model3.predict(X_test.reshape(-1, 1))
```

```
In [ ]: df3 = pd.DataFrame(list(zip(y_test, predict3)), columns=["Actual Values", "Predicted V
```

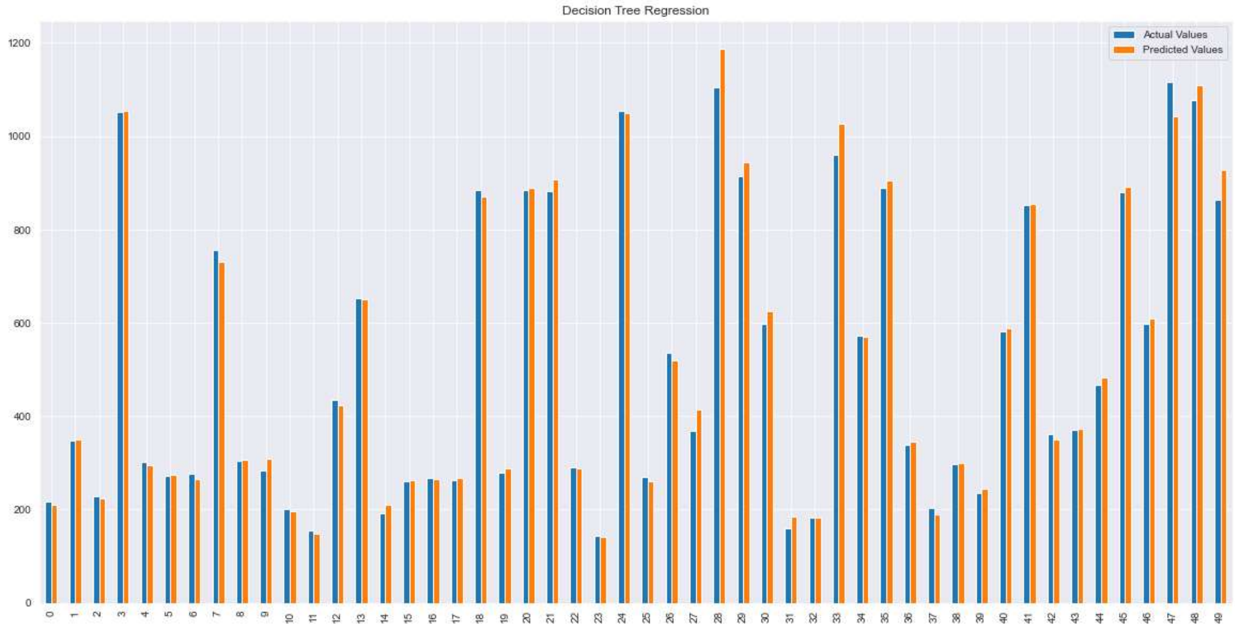
```
In [ ]: df3.head().style.hide_index()
```

Out[ ]:

Actual Values	Predicted Values
215.800000	211.000000
346.600000	350.250000
227.750000	223.500000
1051.550000	1053.450000
301.150000	294.650000

In [ ]:

```
df3.head(50).plot(kind="bar", figsize=(20, 10), title='Decision Tree Regression')
plt.show()
```



In [ ]:

```
accuracy3 = r2_score(y_test, predict3)
print("Accuracy of Decision Tree Regression:", accuracy3)
```

```
Accuracy of Decision Tree Regression: 0.9972340463693731
```

Random Forest Regression

In [ ]:

```
model4 = RandomForestRegressor(n_estimators=100)
build4 = model4.fit(X_train.reshape(-1, 1), y_train)
predict4 = model4.predict(X_test.reshape(-1, 1))
```

In [ ]:

```
df4 = pd.DataFrame(list(zip(y_test, predict4)), columns=["Actual Values", "Predicted V
```

In [ ]:

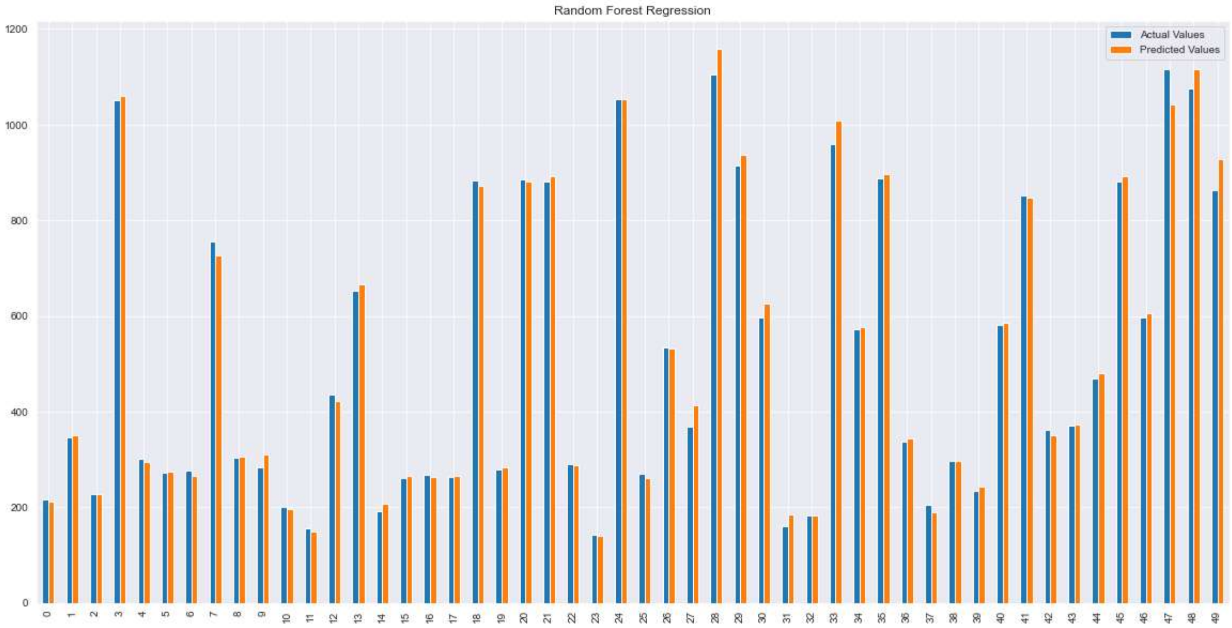
```
df4.head().style.hide_index()
```

Out[ ]:

Actual Values	Predicted Values
215.800000	211.256888
346.600000	351.039500
227.750000	226.917500
1051.550000	1060.292167
301.150000	293.659962

In [ ]:

```
df4.head(50).plot(kind="bar", figsize=(20, 10), title='Random Forest Regression')
plt.show()
```



In [ ]:

```
accuracy4 = r2_score(y_test, predict4)
print("Accuracy of Random Forest Regression:", accuracy4)
```

```
Accuracy of Random Forest Regression: 0.9977574389927836
```

Visualize the results

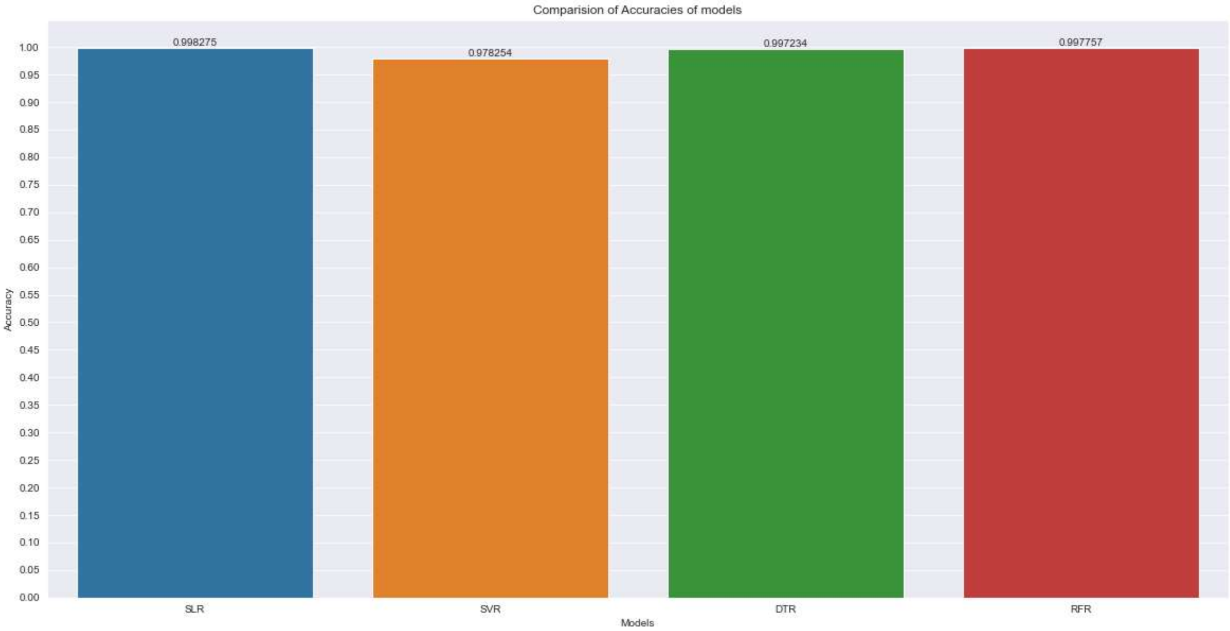
In [ ]:

```
dict1 = {
    "Model": ["Simple Linear Regression", "Support Vector Regression", "Decision Tree
    "Accuracy": np.array([accuracy1, accuracy2, accuracy3, accuracy4])
}
df = pd.DataFrame(dict1)
display(df.style.hide_index())
```

Model	Accuracy
Simple Linear Regression	0.998275
Support Vector Regression	0.978254
Decision Tree Regression	0.997234
Random Forest Regression	0.997757

In [ ]:

```
models = ['SLR', 'SVR', 'DTR', 'RFR']
acc = [accuracy1, accuracy2, accuracy3, accuracy4]
plt.figure(figsize=(20, 10))
plt.title('Comparision of Accuracies of models')
plt.yticks(np.linspace(0,1,21))
plt.ylabel("Accuracy")
plt.xlabel("Models")
values = df.Accuracy
plot = sns.barplot(x=models, y=acc, data=values, errwidth=0)
plot.bar_label(plot.containers[0])
plt.show()
```



Find out the closing price of the company of that day

In [ ]:

```
new_dict = {
    'Date': np.array(['11-May-22']),
    'Open':np.array([718.00])}

future_stock_value = pd.DataFrame(new_dict)
display(future_stock_value.style.hide_index())
```

```
     Date        Open
11-May-22  718.000000
```

Predict using the highest accuracy model

In [ ]:

```
models = np.array(df['Model'])
accuracy = np.array(df['Accuracy'])
```

In [ ]:

```
highest_accuracy=0.0
best_model=""
```

In [ ]:

```
for i in range(len(accuracy)) :
    if accuracy[i] >= highest_accuracy :
        highest_accuracy=accuracy[i]
        best_model=models[i]
```

In [ ]:

```
slr, svr, dtr, rfr = [], [], [], []

if best_model == models[0] :
    future_stock_value['Predicted'] = model1.predict(future_stock_value.Open.values.re
elif best_model == models[1] :
    future_stock_value['Predicted'] = model2.predict(future_stock_value.Open.values.re
elif best_model == models[2] :
    future_stock_value['Predicted'] = model3.predict(future_stock_value.Open.values.re
elif best_model == models[3] :
    future_stock_value['Predicted'] = model4.predict(future_stock_value.Open.values.re
```

In [ ]:

```
display(future_stock_value.style.hide_index())
```

Date	Open	Predicted
11-May-22	718.000000	717.346475

In [ ]: