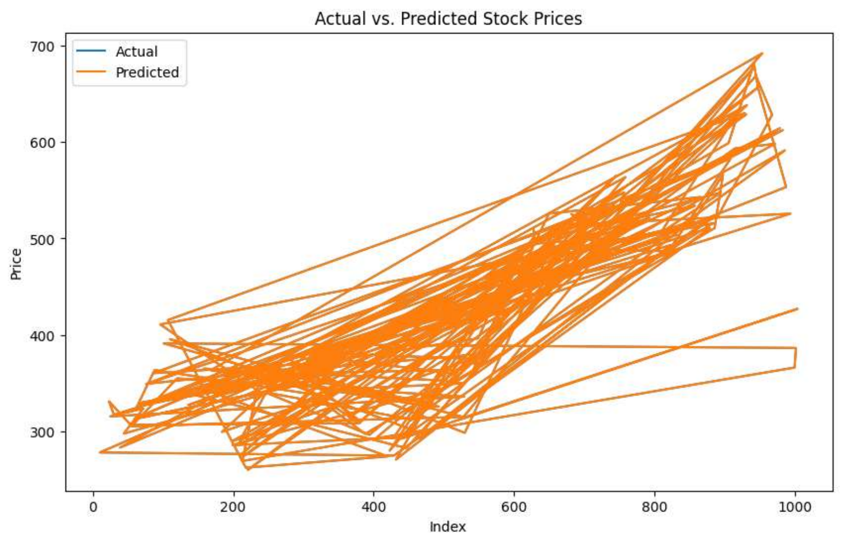
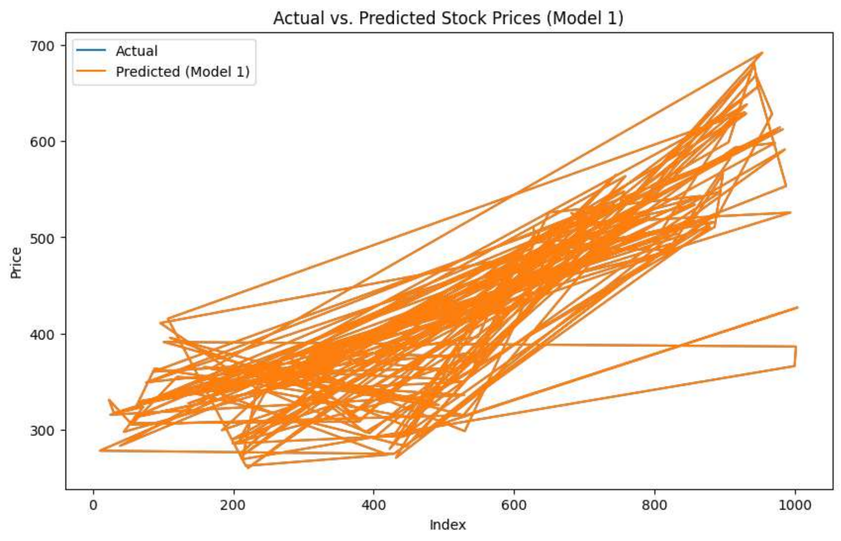
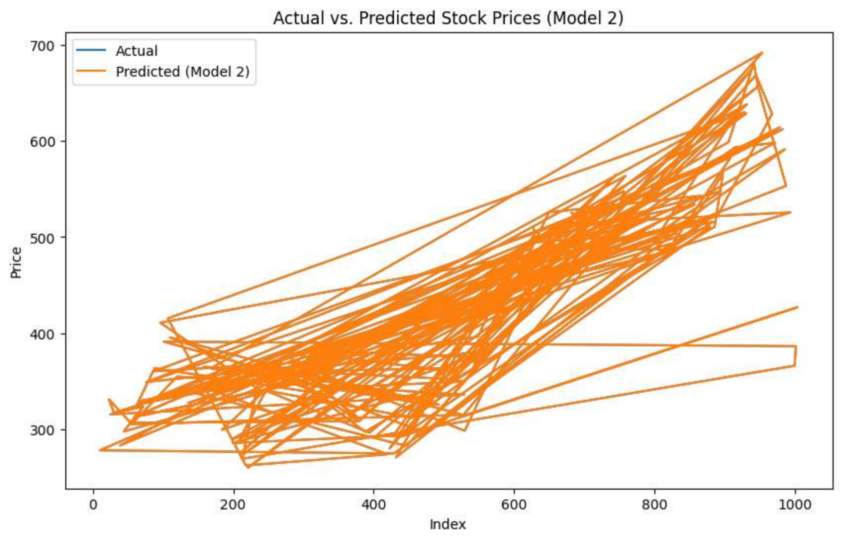
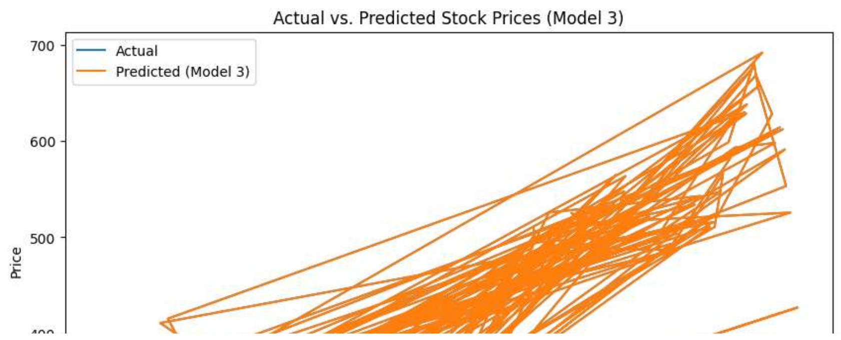
```
# -*- coding: utf-8 -*-
"""StockPricePredictionipynb
Automatically generated by
Colaboratory.
Original file is located at

https://colab.research.google.com/drive/1RCyUb46hNqq_q2jZnxhhEXOz6W9bc_9z#scrollTo=mQbHR8rYYx6c&line=17&uniqifier=3
"""
#Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# %matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
df = pd.read_csv('/content/Stock_Price_data_set_.csv')
df.columns
X = df[['Open', 'High', 'Low', 'Adj Close', 'Volume']]
y = df['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model =  LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
mse
new_data = pd.DataFrame({
 'Open': [262],

'High': [67.899994],
 'Low': [250.029999],
 'Adj Close': [254.259995],
 'Volume':
[11896100]
})
predictions = model.predict(new_data)
predictions
df_preds = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_preds
# Plot the graph for Actual vs.Predicted values
plt.figure(figsize=(10, 6))
plt.plot(df_preds.index, df_preds['Actual'], label='Actual')
plt.plot(df_preds.index, df_preds['Predicted'], label='Predicted')
plt.xlabel('Index')
plt.ylabel('Price')
plt.title('Actual vs. Predicted Stock Prices')
plt.legend()
plt.show()
# Create three Linear Regression models and fit them to the training data
model1 = LinearRegression()
model1.fit(X_train, y_train)
model2 = LinearRegression()
model2.fit(X_train, y_train)
model3 = LinearRegression()
model3.fit(X_train, y_train)
# Make predictions on the test set using the three models
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)
y_pred3 = model3.predict(X_test)
# Create DataFrames with actual and predicted values for the test set
df_preds1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred1})
df_preds2 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred2})
df_preds3 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred3})
# Plot the graphs for Actual vs. Predicted values for the three models
plt.figure(figsize=(10, 6))
plt.plot(df_preds1.index, df_preds1['Actual'],label='Actual')
plt.plot(df_preds1.index, df_preds1['Predicted'], label='Predicted (Model 1)')
plt.xlabel('Index')
plt.ylabel('Price')
plt.title('Actual vs. Predicted Stock Prices (Model 1)')
plt.legend()
plt.show()
plt.figure(figsize=(10, 6))
plt.plot(df_preds2.index,df_preds2['Actual'], label='Actual')
plt.plot(df_preds2.index, df_preds2['Predicted'],label='Predicted (Model 2)')
plt.xlabel('Index')
plt.ylabel('Price')
plt.title('Actual vs. Predicted Stock Prices (Model 2)')
plt.legend()
plt.show()
plt.figure(figsize=(10,
6))
plt.plot(df_preds3.index, df_preds3['Actual'], label='Actual')
plt.plot(df_preds3.index,df_preds3['Predicted'], label='Predicted (Model 3)')
plt.xlabel('Index')
plt.ylabel('Price')
plt.title('Actual vs. Predicted Stock Prices (Model 3)')
plt.legend()
plt.show()
"""Visualization of
model"""
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
# Import RandomForestRegressor
from sklearn.metrics import mean_squared_error
# Select the features and target variable
X = df[['Open', 'High', 'Low', 'Adj Close', 'Volume']]
y = df['Close']
#Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create and fit the models
model1 = LinearRegression()
model1.fit(X_train, y_train)
model2 = RandomForestRegressor()
model2.fit(X_train, y_train)
# Make predictions on the test set
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)
# Calculate the mean squared error (MSE)
mse1 = mean_squared_error(y_test, y_pred1)
mse2 = mean_squared_error(y_test, y_pred2)
# Create a DataFrame with model names and MSE values
model_names = ['Linear Regression', 'Random Forest']
mse_values = [mse1, mse2]
df_mse = pd.DataFrame({'Model': model_names, 'MSE': mse_values})
# Plot the MSE values
plt.figure(figsize=(8, 6))
plt.bar(df_mse['Model'],df_mse['MSE'])
plt.xlabel('Model')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Comparison of Models')
plt.show()
# Plot the scatter plot of Actual vs. Predicted values for each model
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred1, label='Linear Regression')
plt.scatter(y_test, y_pred2, label='Random Forest')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs. Predicted Values')
plt.legend()
plt.show()
"""Validation of
model1"""
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
# Read the CSV file into a pandas DataFrame
df = pd.read_csv('/content/Stock_Price_data_set_.csv')
# Select the features and target variable
X = df[['Open', 'High', 'Low', 'Adj Close', 'Volume']]
y = df['Close']
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Create a Linear Regression model and fit it to the training data
model1 = LinearRegression()
model1.fit(X_train, y_train)
# Make predictions on the test set
```

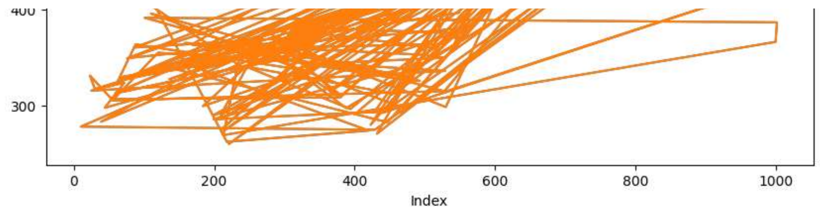
```
y_pred = model1.predict(X_test)
# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Print the evaluation metrics
print("Mean Squared Error(MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Coefficient of Determination (R^2):", r2)
# Calculate the sum of errors
errors = y_test -y_pred
sum_of_errors = errors.sum()
print("Sum of Errors:", sum_of_errors)
#Calculate the sum of squared errors
mse = mean_squared_error(y_test, y_pred)
sse = mse * len(y_test)
print("Sum of Squared Errors (SSE):", sse)
"""Validation
of model2"""
# Create a Random Forest model and fit it to the training data
model2 = RandomForestRegressor()
model2.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model2.predict(X_test)
# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Print the evaluation metrics
print("Mean Squared Error(MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Coefficient of Determination (R^2):", r2)
# Calculate the SSE
sse = np.sum((y_test - y_pred)**2)
#Print the SSE
print("Sum of Squared Errors (SSE):",sse)
"""Validation of model 3"""
# Create a new model (model3) and fit it to the training data
# Replace 'model3' with the appropriate model you want to validate
model3 = LinearRegression()
model3.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model3.predict(X_test)
# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 =  r2_score(y_test, y_pred)
# Print the evaluation metrics
print("Mean Squared Error(MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Coefficient of Determination (R^2):", r2)
# Calculate the SSE
sse = np.sum((y_test - y_pred) ** 2)
#Print the SSE
print("Sum of Squared Errors (SSE):", sse)
"""Building
best fit manually"""
# Train and evaluate different models
models = [

LinearRegression(),
 # Add more models you want to evaluate here
]
best_model =None
best_mse = float('inf')
for model in models:
 # Train the model
  model.fit(X_train, y_train)
 # Make predictions on the test set
  y_pred = model.predict(X_test)
 # Calculate mean squared error (MSE)
  mse = mean_squared_error(y_test, y_pred)
 # Check if this model has the lowest MSE
  if mse < best_mse:
    best_mse = mse
    best_model = model
# Print the best model and its MSE
print("Best Model:", best_model)
print("Best Model:", best_model)
print("MSE:", best_mse)
# Train the best-fit model
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Calculate mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)
# Plot the predicted values against the actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values (Best-fit Model)\nMSE: {:.2f}".format(mse))
plt.show()
# Calculate the SSE
sse = mse * len(y_test)
# Print the SSE
print("Sum of Squared Errors (SSE):", sse)
#Creating a linear regression model
model = LinearRegression()
# Fit the model to the data
model.fit(X, y)
# Fetch the intercept (b1)
intercept = model.intercept_
# Fetch the coefficients (b2)
coefficients = model.coef_
print("Intercept (b1):",intercept)
print("Coefficients (b2):", coefficients)
```

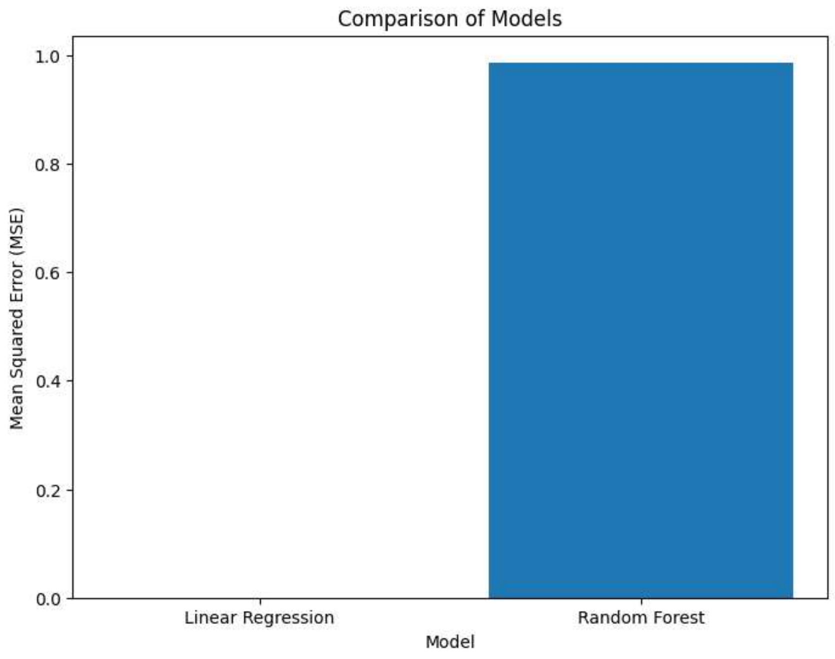

Actual vs. Predicted Stock Prices
Actual
Predicted
Price
Index
700
600
500
400
300
0
200
400
600
800
1000

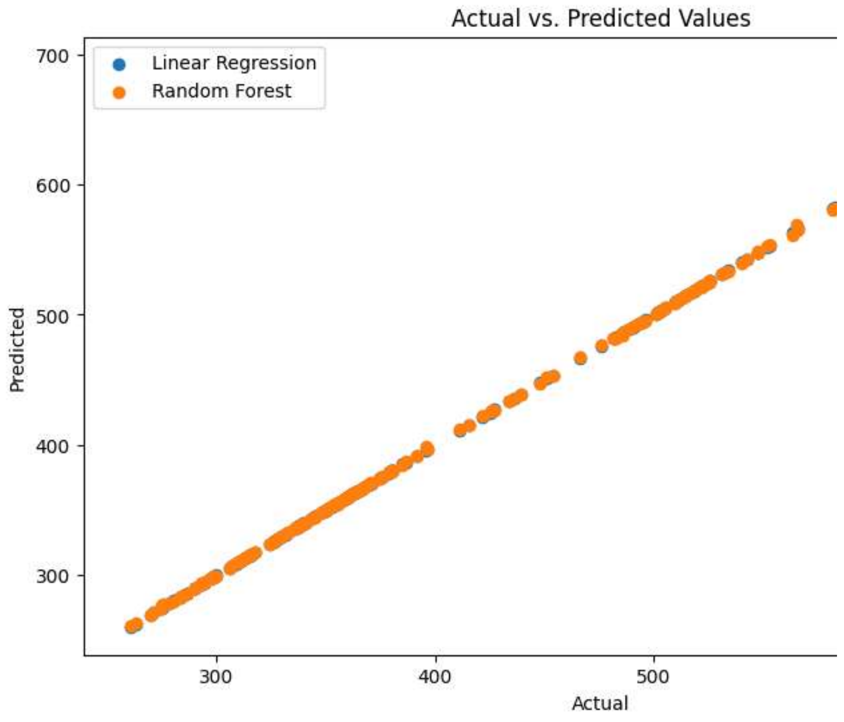

Actual vs. Predicted Stock Prices (Model 1)
Actual
Predicted (Model 1)
Price
Index
700
600
500
400
300
0
200
400
600
800
1000


Actual vs. Predicted Stock Prices (Model 2)
Actual
Predicted (Model 2)
Price
Index
700
600
500
400
300
0
200
400
600
800
1000

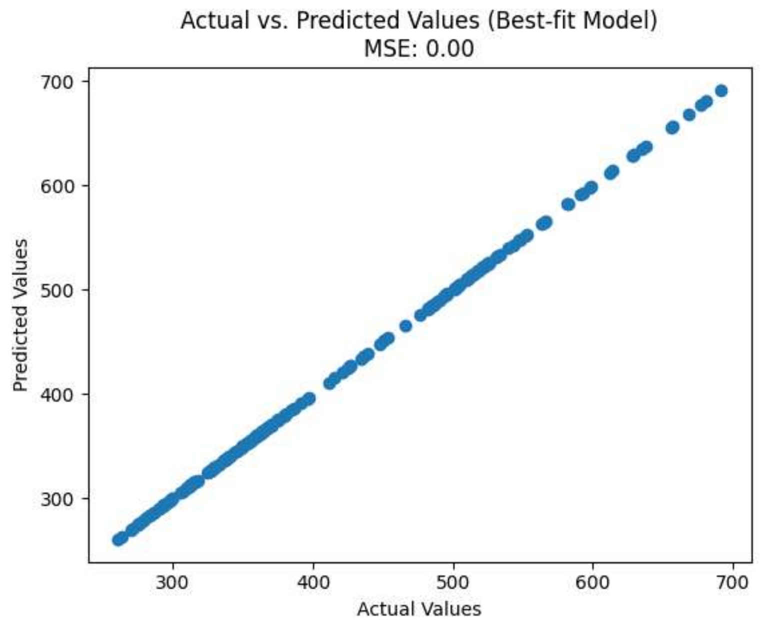

Actual vs. Predicted Stock Prices (Model 3)
Actual
Predicted (Model 3)
Price
700
600
500







```
Mean Squared Error(MSE): 3.3319485147508446e-26
Mean Absolute Error (MAE): 1.5449028195338417e-13
Coefficient of Determination (R^2): 1.0
Sum of Errors: 8.526512829121202e-13
Sum of Squared Errors (SSE): 6.730535999796706e-24
Mean Squared Error(MSE): 0.7250609646843547
Mean Absolute Error (MAE): 0.45416893163366717
Coefficient of Determination (R^2): 0.9999378166970622
Sum of Squared Errors (SSE): 146.46231486623964
Mean Squared Error(MSE): 3.3319485147508446e-26
Mean Absolute Error (MAE): 1.5449028195338417e-13
Coefficient of Determination (R^2): 1.0
Sum of Squared Errors (SSE): 6.730535999796706e-24
Best Model: LinearRegression()
MSE: 3.3319485147508446e-26
```



```
Sum of Squared Errors (SSE): 6.730535999796706e-24
Intercept (b1): -7.389644451905042e-13
Coefficients (b2): [-4.44124313e-15  1.34218130e-15  3.89672650e-15  1.00000000e+0
  2.71050543e-20]
```

Colab paid products - Cancel contracts here

✓ 4s completed at 10:18 PM