```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
```

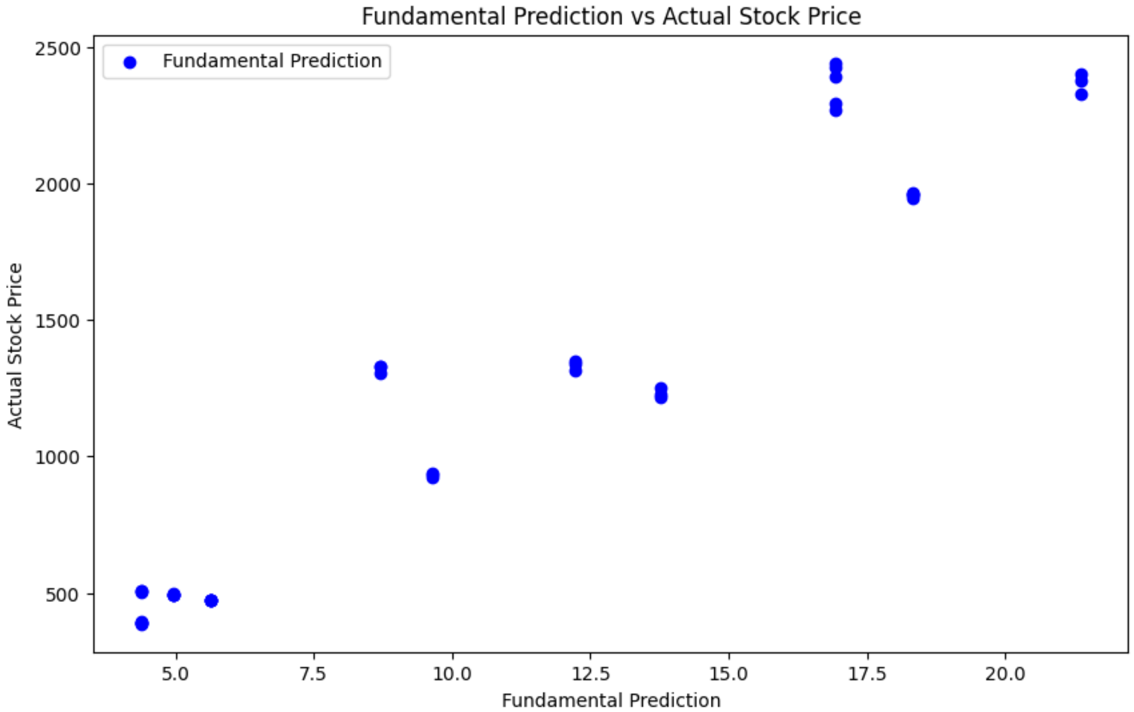
```
df = pd.read_csv('/content/fusiondataset.csv')
```

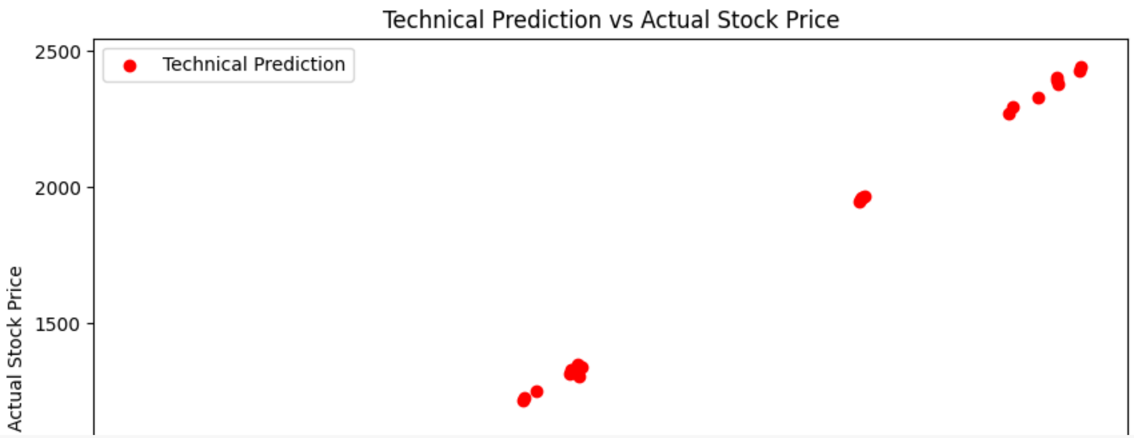
```
df
```

	EPS Prediction	Closing Price Predicted	Actual stock price
0	9.631759	919.67	921.812134
1	9.631759	931.62	932.015442
2	9.631759	939.16	938.751587
3	12.224196	1339.67	1347.427734
4	12.224196	1348.04	1338.958008
5	12.224196	1322.02	1314.886230
6	13.757649	1219.61	1214.537354
7	13.757649	1222.72	1226.028442
8	13.757649	1250.39	1252.923584
9	4.389111	519.13	509.767914
10	4.389111	501.49	501.917328
11	4.389111	505.19	505.161560
12	21.375632	2381.94	2398.550049
13	21.375632	2384.83	2378.300049
14	21.375632	2340.09	2325.550049
15	8.699729	1342.30	1303.791382
16	8.699729	1328.10	1330.240601
17	8.699729	1325.92	1327.120239
18	5.646127	473.94	472.322845

```python
import matplotlib.pyplot as plt

# Assuming 'Fundamental_Prediction', 'Technical_Prediction', and 'Actual_Stock_Price' are column names
fundamental_predictions = df['EPS Prediction']
technical_predictions = df['Closing Price Predicted']
actual_stock_price = df['Actual stock price']

# Scatter plot for Fundamental Prediction vs Actual Stock Price
plt.figure(figsize=(10, 6))
plt.scatter(fundamental_predictions, actual_stock_price, color='blue', label='Fundamental Prediction')
plt.xlabel('Fundamental Prediction')
plt.ylabel('Actual Stock Price')
plt.title('Fundamental Prediction vs Actual Stock Price')
plt.legend()
plt.show()

# Scatter plot for Technical Prediction vs Actual Stock Price
plt.figure(figsize=(10, 6))
plt.scatter(technical_predictions, actual_stock_price, color='red', label='Technical Prediction')
plt.xlabel('Technical Prediction')
plt.ylabel('Actual Stock Price')
plt.title('Technical Prediction vs Actual Stock Price')
```

```
plt.legend()
plt.show()

# Line plot for Fundamental Prediction, Technical Prediction, and Actual Stock Price
plt.figure(figsize=(10, 6))
plt.plot(fundamental_predictions, label='Fundamental Prediction')
plt.plot(technical_predictions, label='Technical Prediction')
plt.plot(actual_stock_price, label='Actual Stock Price')
plt.xlabel('Data Point')
plt.ylabel('Price')
plt.title('Predictions vs Actual Stock Price')
plt.legend()
plt.show()
```






```
# Assuming 'Fundamental_Prediction' and 'Technical_Prediction' are column names in the CSV
fundamental_predictions = df['EPS Prediction'].values.reshape(-1, 1)
technical_predictions = df['Closing Price Predicted'].values.reshape(-1, 1)
actual_targets = df['Actual stock price'].values.reshape(-1, 1)  # Replace with your actual target column name
```

500

```
X  = df[['EPS Prediction', 'Closing Price Predicted']]
y  = df['Actual stock price']
```

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    np.column_stack((fundamental_predictions, technical_predictions)),
    actual_targets, test_size=0.2, random_state=42)
```

```
# Train base models
fundamental_model = RandomForestRegressor()
fundamental_model.fit(fundamental_predictions, actual_targets)
technical_model = MLPRegressor(hidden_layer_sizes=(10,10))  # Adjust hidden layer sizes as needed
technical_model.fit(technical_predictions, actual_targets)
```

```
<ipython-input-23-4e190c5fadfe>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  fundamental_model.fit(fundamental_predictions, actual_targets)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:1623: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the s
  y = column_or_1d(y, warn=True)
▾           MLPRegressor
MLPRegressor(hidden_layer_sizes=(10, 10))
```

```
# Obtain predictions from base models
fundamental_predictions_train = fundamental_model.predict(fundamental_predictions)
technical_predictions_train = technical_model.predict(technical_predictions)
```

```
# Stack the predictions for training the meta-model
stacked_features_train = np.column_stack((fundamental_predictions_train, technical_predictions_train))
```

Data Point

```
# Train the meta-model (Neural Network)
meta_model = Sequential([
    Dense(10, activation='relu', input_dim=2),  # Adjust the number of neurons as needed
    Dense(1)  # Output layer
])
```

```
meta_model.compile(optimizer='adam', loss='mean_squared_error')
meta_model.fit(stacked_features_train, actual_targets, epochs=1000, batch_size=32)
```

```
2/2 [==============================] - 0s 6ms/step - loss: 335.3829
Epoch 741/1000
2/2 [==============================] - 0s 7ms/step - loss: 337.2167
Epoch 742/1000
2/2 [==============================] - 0s 8ms/step - loss: 338.8760
Epoch 743/1000
2/2 [==============================] - 0s 8ms/step - loss: 337.3434
Epoch 744/1000
2/2 [==============================] - 0s 9ms/step - loss: 334.6840
Epoch 745/1000
2/2 [==============================] - 0s 7ms/step - loss: 334.6488
Epoch 746/1000
2/2 [==============================] - 0s 11ms/step - loss: 334.8601
Epoch 747/1000
2/2 [==============================] - 0s 6ms/step - loss: 336.4365
Epoch 748/1000
2/2 [==============================] - 0s 6ms/step - loss: 337.3625
Epoch 749/1000
2/2 [==============================] - 0s 6ms/step - loss: 332.3364
Epoch 750/1000
2/2 [==============================] - 0s 6ms/step - loss: 330.1559
Epoch 751/1000
2/2 [==============================] - 0s 14ms/step - loss: 349.9369
Epoch 752/1000
2/2 [==============================] - 0s 6ms/step - loss: 349.8973
Epoch 753/1000
2/2 [==============================] - 0s 7ms/step - loss: 339.0022
Epoch 754/1000
2/2 [==============================] - 0s 5ms/step - loss: 349.4843
Epoch 755/1000
2/2 [==============================] - 0s 7ms/step - loss: 334.5739
Epoch 756/1000
2/2 [==============================] - 0s 9ms/step - loss: 332.8574
Epoch 757/1000
2/2 [==============================] - 0s 10ms/step - loss: 336.2375
Epoch 758/1000
2/2 [==============================] - 0s 12ms/step - loss: 332.9759
Epoch 759/1000
2/2 [==============================] - 0s 8ms/step - loss: 337.2145
```

```
# Obtain predictions from base models on the test set
fundamental_predictions_test = fundamental_model.predict(X_test[:, 0].reshape(-1, 1))
technical_predictions_test = technical_model.predict(X_test[:, 1].reshape(-1, 1))
```

```
# Stack the predictions for the test set
stacked_features_test = np.column_stack((fundamental_predictions_test, technical_predictions_test))
```

```
# Predict using the meta-model (Neural Network)
combined_predictions_test = meta_model.predict(stacked_features_test)
```

```
1/1 [==============================] - 0s 49ms/step
```

```
from sklearn.metrics import mean_squared_error, r2_score
# Evaluate the performance of the combined predictions
```

```
mse_combined = mean_squared_error(y_test, combined_predictions_test)
print('Mean Squared Error for combined predictions:', mse_combined)
r2_combined = r2_score(y_test, combined_predictions_test)
print('R-squared for combined predictions:', r2_combined)
```

```
Mean Squared Error for combined predictions: 116.27612800721651
R-squared for combined predictions: 0.9997701038081679
```

```
print('Combined Prediction: ',combined_predictions_test)
print('Actual :',y_test)
```

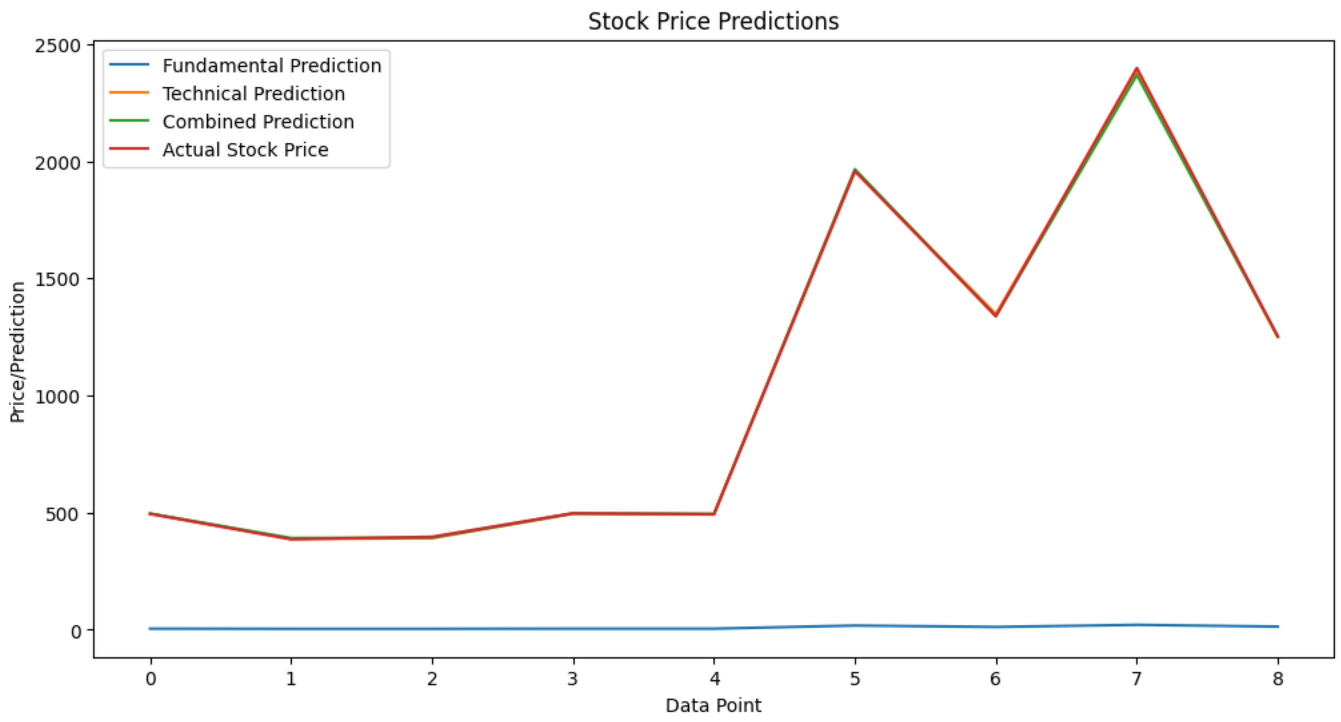
```
Combined Prediction:  [[ 496.353  ]
 [ 392.00418]
 [ 393.11484]
 [ 496.3143 ]
 [ 495.4165 ]
 [1966.129  ]
 [1338.3401 ]
 [2368.241  ]
 [1250.9573 ]]
Actual : [[ 494.884003]
 [ 386.436951]
 [ 397.234619]
 [ 497.880585]
 [ 492.704651]
 [1958.150024]
 [1338.958008]
 [2398.550049]
 [1252.923584]]
```

```



# Predict using the meta-model (Neural Network)
combined_predictions_test = meta_model.predict(stacked_features_test)

# Convert the predictions and actual inputs to a DataFrame for easy visualization
combined_predictions_df = pd.DataFrame({
    'Fundamental_Prediction': X_test[:, 0],  # Assuming 0 is the index for EPS prediction
    'Technical_Prediction': X_test[:, 1],  # Assuming 1 is the index for closing price prediction
    'Combined_Prediction': combined_predictions_test.flatten(),
    'Actual_Stock_Price': y_test.flatten()
})

# Print the DataFrame to view the predictions and actual inputs
print(combined_predictions_df)
```

```
1/1 [==============================] - 0s 21ms/step
   Fundamental_Prediction  Technical_Prediction  Combined_Prediction  \
0                4.950175                495.80           496.352997
1                4.371347                388.20           392.004181
2                4.371347                391.07           393.114838
3                4.950175                495.70           496.314301
4                4.950175                493.38           495.416504
5               18.328121               1957.98          1966.129028
6               12.224196               1348.04          1338.340088
7               21.375632               2381.94          2368.240967
```

```
   8          13.757649          1250.39       1250.957275

      Actual_Stock_Price
   0          494.884003
   1          386.436951
   2          397.234619
   3          497.880585
   4          492.704651
   5         1958.150024
   6         1338.958008
   7         2398.550049
   8         1252.923584
```

```
import matplotlib.pyplot as plt


# Convert the predictions and actual inputs to a DataFrame for easy visualization
combined_predictions_df = pd.DataFrame({
    'Fundamental_Prediction': X_test[:, 0].flatten(),  # Assuming 0 is the index for EPS prediction
    'Technical_Prediction': X_test[:, 1].flatten(),  # Assuming 1 is the index for closing price prediction
    'Combined_Prediction': combined_predictions_test.flatten(),
    'Actual_Stock_Price': y_test.flatten()
})

# Plot the predictions
plt.figure(figsize=(12, 6))
plt.plot(combined_predictions_df['Fundamental_Prediction'], label='Fundamental Prediction')
plt.plot(combined_predictions_df['Technical_Prediction'], label='Technical Prediction')
plt.plot(combined_predictions_df['Combined_Prediction'], label='Combined Prediction')
plt.plot(combined_predictions_df['Actual_Stock_Price'], label='Actual Stock Price')

plt.xlabel('Data Point')
plt.ylabel('Price/Prediction')
plt.title('Stock Price Predictions')
plt.legend()
plt.show()
```


Stock Price Predictions
Fundamental Prediction
Technical Prediction
Combined Prediction
Actual Stock Price
Price/Prediction
Data Point
0
500
1000
1500
2000
2500
0
1
2
3
4
5
6
7
8