Enhancing Stock Price Prediction: Unveiling the Power of Feature Engineering, Model Training, and Evaluation

Introduction

Welcome to the presentation on *Enhancing Stock Price Prediction: Unveiling the Power of Feature Engineering, Model Training, and Evaluation*. This presentation explores the key techniques and approaches to improve stock price prediction accuracy. We will delve into the importance of feature engineering, effective model training, and robust evaluation methodologies. Let's get started!

Feature Engineering

Feature engineering plays a crucial role in stock price prediction. It involves transforming raw data into meaningful features that capture relevant information. Techniques like *lagging indicators*, *moving averages*, and *technical analysis indicators* can be employed to extract valuable insights from historical stock data. Effective feature engineering enables models to learn and leverage patterns in the data for accurate predictions.

Model Training

Model training is a critical step in stock price prediction. Various machine learning algorithms such as *linear regression*, *random forest*, and *LSTM* can be employed to train predictive models. The models learn from historical data and aim to capture underlying patterns and trends. Proper parameter tuning, regularization techniques, and model selection are essential to optimize model performance.

Evaluation Metrics

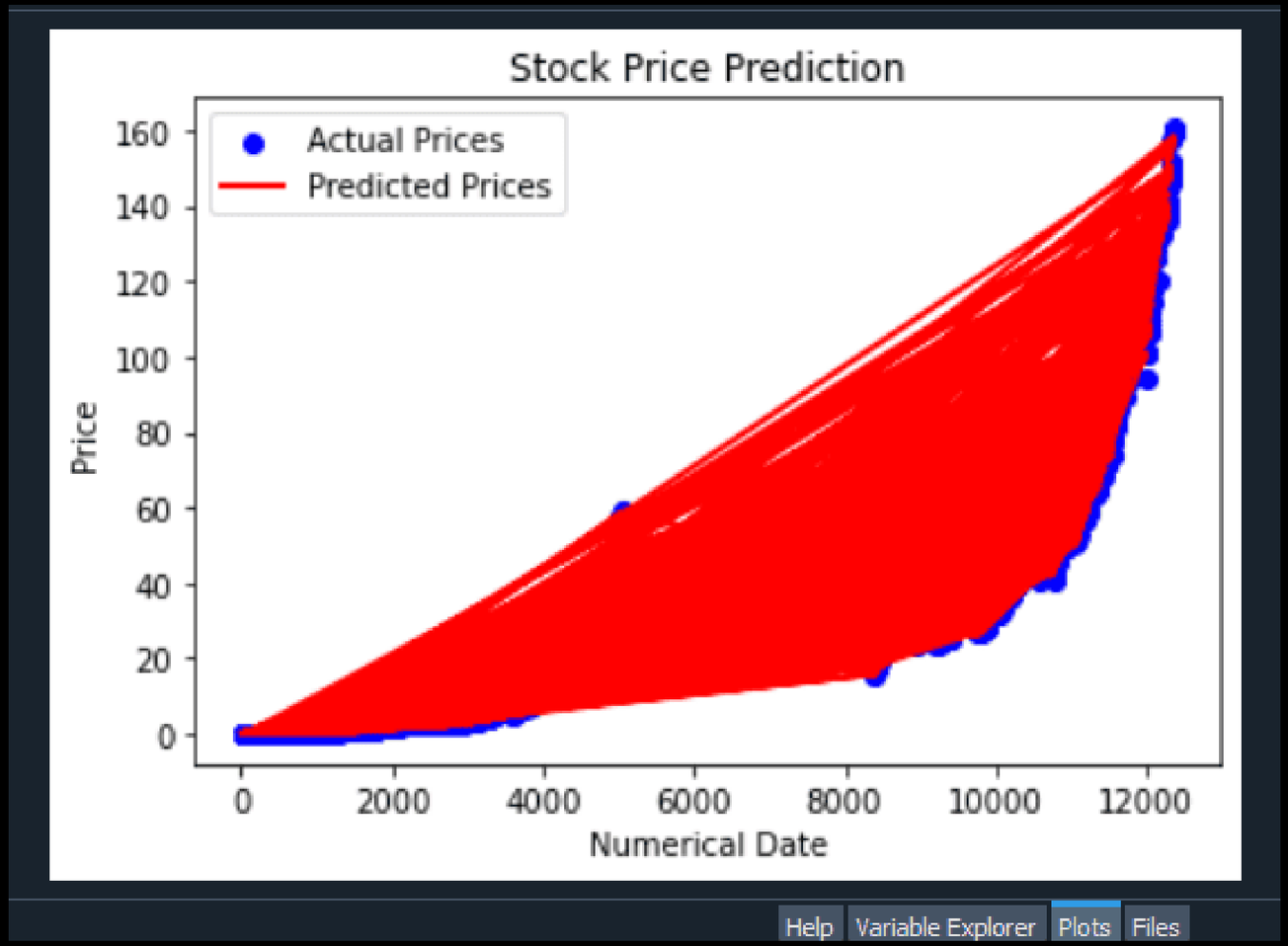
Evaluating the performance of stock price prediction models is crucial. Common evaluation metrics include *mean absolute error (MAE)*, *root mean square error (RMSE)*, and *mean absolute percentage error (MAPE)*. These metrics help quantify the accuracy of predictions and assess the model's ability to capture stock price movements. A comprehensive evaluation framework ensures reliable and robust predictions.

Advanced Techniques

In addition to feature engineering and model training, advanced techniques can further enhance stock price prediction. These techniques include *ensemble learning*, *deep learning*, and *reinforcement learning*. Ensemble learning combines multiple models to improve prediction accuracy, while deep learning leverages neural networks for complex pattern recognition. Reinforcement learning enables models to learn optimal trading strategies.

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load historical stock price data from a CSV file
data = pd.read_csv("C:/Users/balur/Downloads/MSFT.csv")  # Replace with your data source

# Assuming the dataset has 'Date' and 'Close' columns
X = data['Date'].values
y = data['Close'].values

# Feature Engineering (Simple date processing and creating a numerical date feature)
data['Date'] = pd.to_datetime(data['Date'])
data['NumericalDate'] = data['Date'].sub(data['Date'].min()).dt.days

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['NumericalDate'].values, y, test_size=0.2, random_state=42)

# Create and train a machine learning model (Random Forest Regressor)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train.reshape(-1, 1), y_train)

# Predict stock prices for the test set
y_pred = model.predict(X_test.reshape(-1, 1))

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
# Assuming the dataset has 'Date' and 'Close' columns
X = data['Date'].values
y = data['Close'].values

# Feature Engineering (Simple date processing and creating a numerical date feature)
data['Date'] = pd.to_datetime(data['Date'])
data['NumericalDate'] = data['Date'].sub(data['Date'].min()).dt.days

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['NumericalDate'].values, y, test_size=0.2, random_state=42)

# Create and train a machine learning model (Random Forest Regressor)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train.reshape(-1, 1), y_train)

# Predict stock prices for the test set
y_pred = model.predict(X_test.reshape(-1, 1))

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Plot the actual vs. predicted prices
plt.scatter(X_test, y_test, color='blue', label='Actual Prices')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted Prices')
plt.title('Stock Price Prediction')
plt.xlabel('Numerical Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```


Stock Price Prediction
Actual Prices
Predicted Prices
160
140
120
100
80
60
40
20
0
Price
0
2000
4000
6000
8000
10000
12000
Numerical Date
Help
Variable Explorer
Plots
Files

```
In [4]: runfile('C:/Users/balur/untitled7.py', wdir=
Mean Absolute Error: 0.3221958684633431 4
Mean Squared Error: 0.3877141951346212
R-squared: 0.9995491811905366

In [5]:
```

IPyt

Conclusion

In this presentation, we explored the power of feature engineering, model training, and evaluation in enhancing stock price prediction. Effective feature engineering helps capture relevant information, while proper model training optimizes predictive performance. Robust evaluation metrics ensure reliable predictions. By leveraging advanced techniques, we can further improve accuracy. Implementing these strategies can empower investors and financial professionals to make informed decisions.