In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the dataset
data = pd.read_csv('NIFTY50_all.csv')
```

In [3]:
```
# Calculate 15-day and 30-day moving averages
data['15_day_ma'] = data['Close'].rolling(window=15).mean()
data['30_day_ma'] = data['Close'].rolling(window=30).mean()
```

In [4]:
```
# Feature Engineering
# Drop rows with NaN values due to the window size of the moving averages
data.dropna(inplace=True)
```

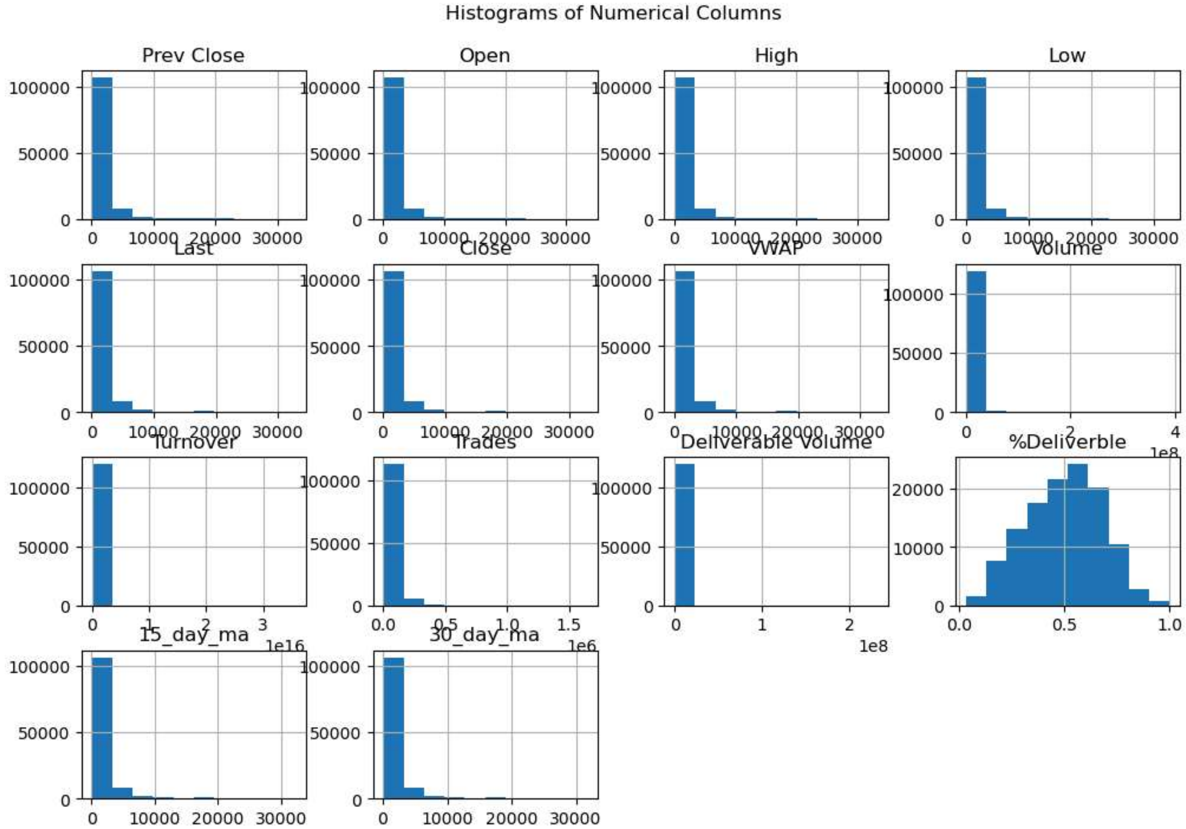
In [42]:
```
# Check the head of the DataFrame
print("Head of the DataFrame:")
print(data.head())
```

```
Head of the DataFrame:
           Date      Symbol Series  Prev Close    Open    High     Low  \
866  2011-06-01  MUNDRAPORT     EQ      161.45  162.10  165.70  161.25
867  2011-06-02  MUNDRAPORT     EQ      164.00  164.00  165.15  160.15
868  2011-06-03  MUNDRAPORT     EQ      161.25  161.50  162.80  159.20
869  2011-06-06  MUNDRAPORT     EQ      161.05  160.50  161.10  159.05
870  2011-06-07  MUNDRAPORT     EQ      159.85  159.85  162.75  156.35

       Last   Close    VWAP   Volume      Turnover   Trades  \
866  163.50  164.00  164.08  2574106  4.223703e+13  19171.0
867  161.15  161.25  162.17  1699298  2.755678e+13  16176.0
868  161.00  161.05  161.02  1185817  1.909361e+13  14810.0
869  160.00  159.85  160.09   546378  8.746905e+12   7071.0
870  157.00  157.25  158.52  2193466  3.477027e+13  17865.0

     Deliverable Volume  %Deliverble   15_day_ma   30_day_ma
866           1271255.0       0.4939  149.103333  144.388333
867            791462.0       0.4658  150.973333  144.835000
868            722154.0       0.6090  152.496667  145.298333
869            386144.0       0.7067  153.906667  145.733333
870           1425849.0       0.6500  154.763333  146.145000
```

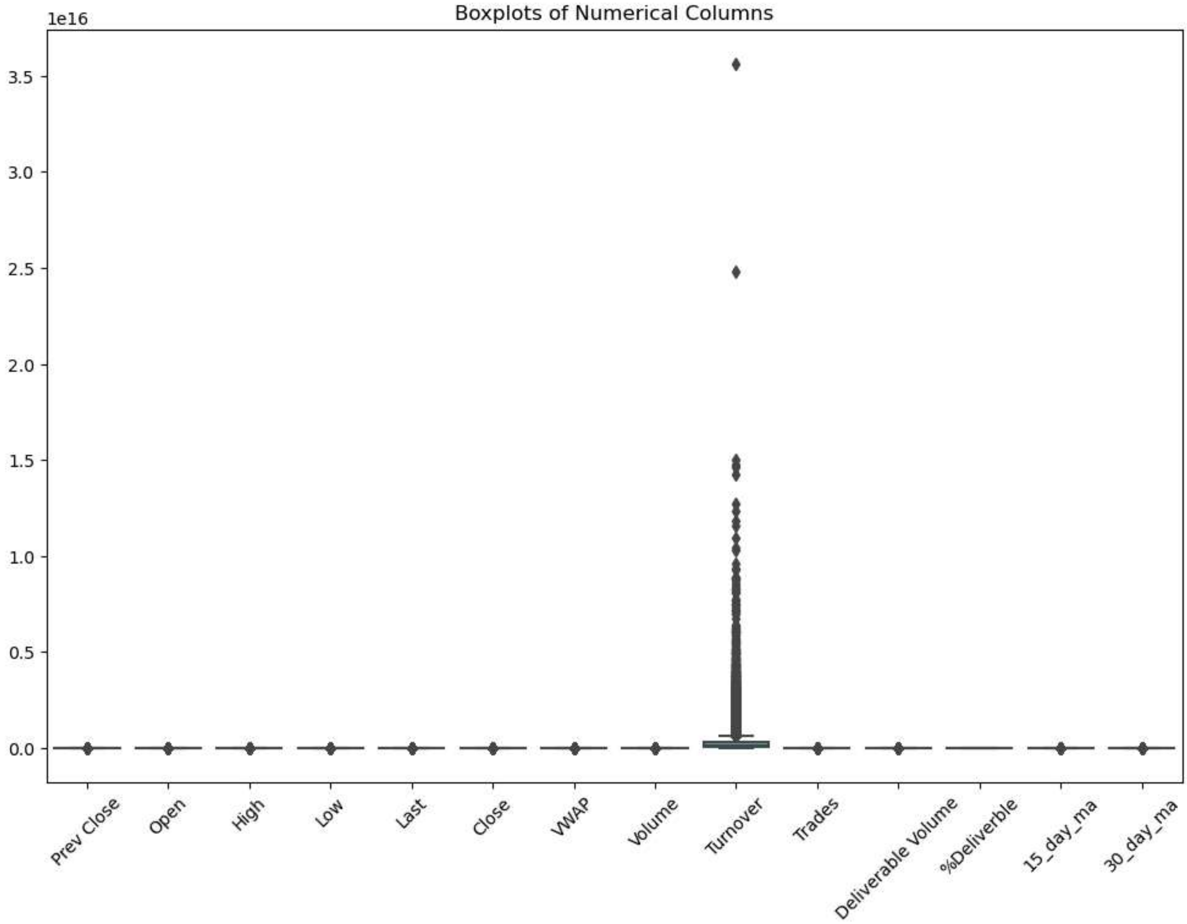
In [43]:
```
# Check the column names
print("\nColumn Names:")
print(data.columns)
```

```
Column Names:
Index(['Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble', '15_day_ma', '30_day_ma'],
      dtype='object')
```

In [44]:
```python
# Histograms of numerical columns
data.hist(figsize=(12, 8))
plt.suptitle('Histograms of Numerical Columns', y=0.95)
plt.show()
```

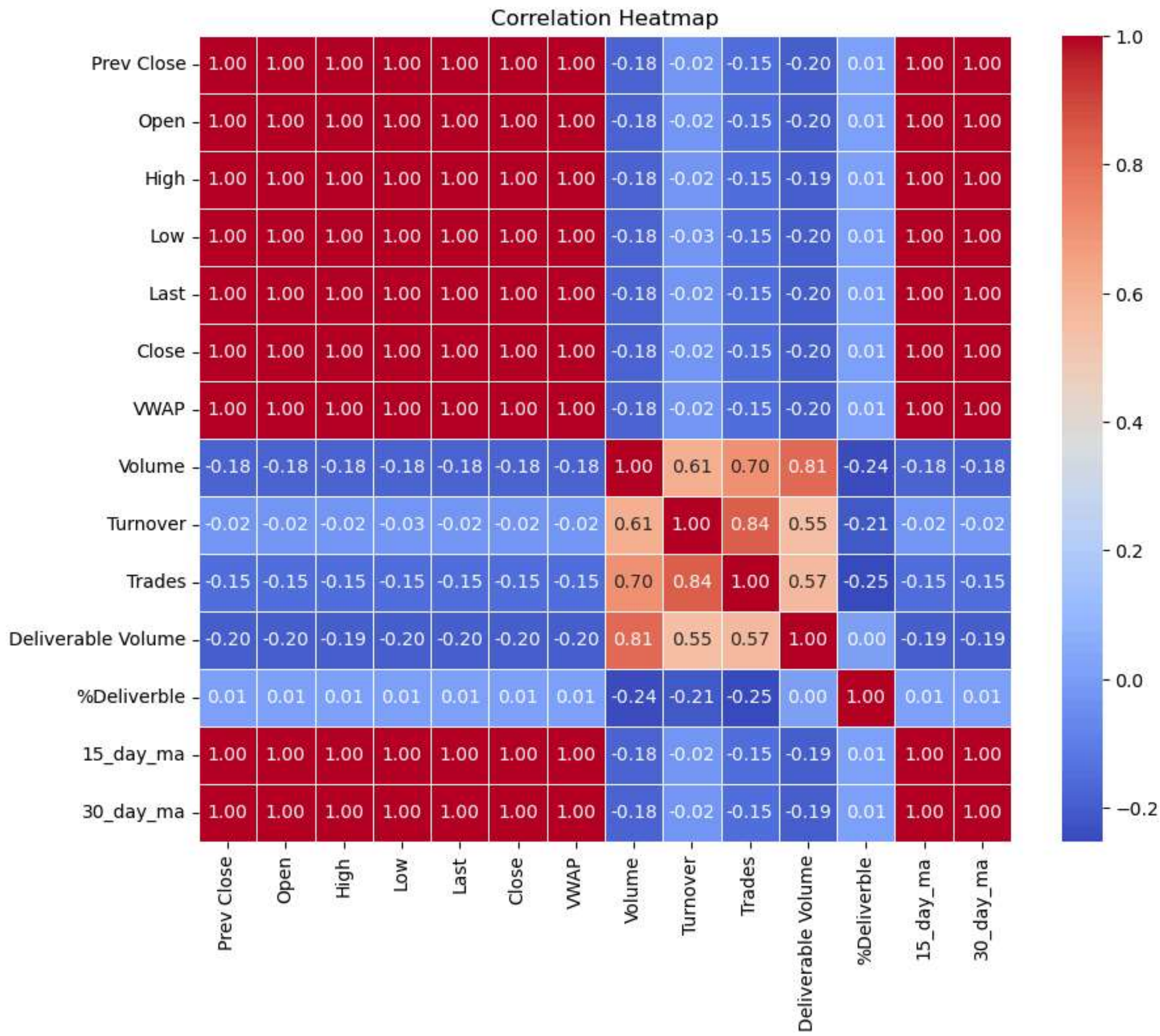


In [45]:
```python
# Boxplots of numerical columns
plt.figure(figsize=(12, 8))
sns.boxplot(data=data)
plt.title('Boxplots of Numerical Columns')
plt.xticks(rotation=45)
plt.show()
```



In [47]:

```
# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f", l
plt.title('Correlation Heatmap')
plt.show()
```



In [5]:
```
# Select features and target variable
features = ['15_day_ma', '30_day_ma']
X = data[features]
y = np.where(data['Close'].shift(-1) > data['Close'], 1, 0)  # 1 if price increases
```

In [6]:
```
# Scaling features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [7]:
```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, rar
```

In [8]:
```
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
print("Random Forest Accuracy:", rf_accuracy)
```

```
Random Forest Accuracy: 0.4962815239519714
```

In [9]:
```
# Support Vector Machine
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
print("SVM Accuracy:", svm_accuracy)
```

```
SVM Accuracy: 0.5057127425318875
```

In [10]:
```
# Gradient Boosting
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)
gb_pred = gb_model.predict(X_test)
gb_accuracy = accuracy_score(y_test, gb_pred)
print("Gradient Boosting Accuracy:", gb_accuracy)
```

```
Gradient Boosting Accuracy: 0.5052141759109228
```

In [11]:
```
# LSTM Model
# Reshape the data for LSTM
X_train_lstm = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test_lstm = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

In [13]:
```
from keras.layers import Input
```

In [14]:
```
# Define the LSTM model
input_shape = (X_train_lstm.shape[1], 1)
model = Sequential()
model.add(Input(shape=input_shape))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))
```

In [15]:
```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [17]:
```
# Train the model
model.fit(X_train_lstm, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 46s 15ms/step - loss: 0.2501
Epoch 2/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 47s 16ms/step - loss: 0.2500
Epoch 3/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 44s 14ms/step - loss: 0.2501
Epoch 4/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 44s 15ms/step - loss: 0.2500
Epoch 5/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 45s 15ms/step - loss: 0.2499
Epoch 6/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 46s 15ms/step - loss: 0.2501
Epoch 7/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 47s 16ms/step - loss: 0.2500
Epoch 8/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 48s 16ms/step - loss: 0.2501
Epoch 9/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 45s 15ms/step - loss: 0.2500
Epoch 10/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 45s 15ms/step - loss: 0.2500
```

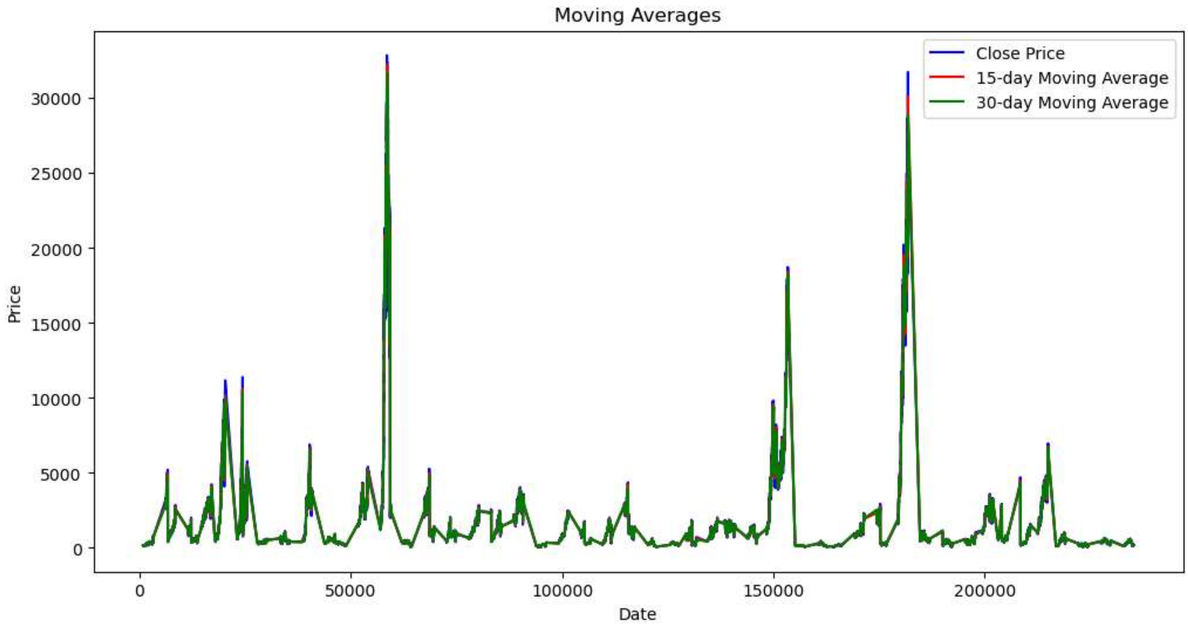
Out[17]:
```
<keras.src.callbacks.history.History at 0x2a397dc1dd0>
```

In [18]:
```
# Evaluate the LSTM model
lstm_pred = model.predict(X_test_lstm)
lstm_pred = (lstm_pred > 0.5)
lstm_accuracy = accuracy_score(y_test, lstm_pred)
print("LSTM Accuracy:", lstm_accuracy)
```

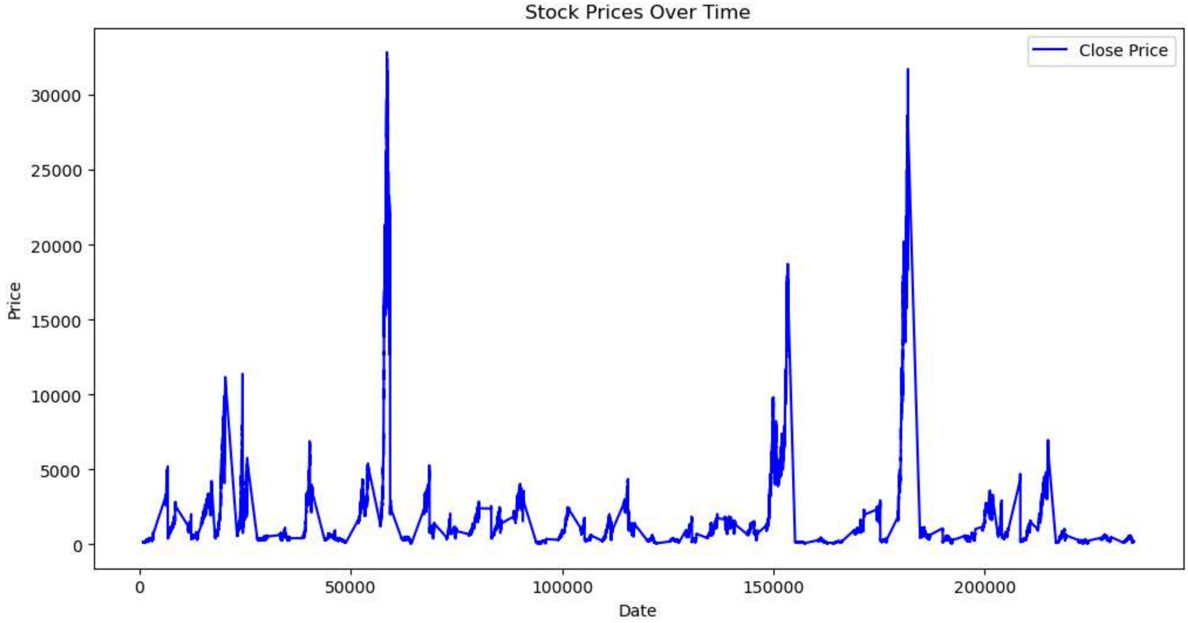
```
753/753 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step
LSTM Accuracy: 0.506543686900162
```

In [19]:
```
# Conclusion
# Summarize findings and suggest the best model for stock market prediction
# Compare the accuracies of all models
models = ['Random Forest', 'SVM', 'Gradient Boosting', 'LSTM']
accuracies = [rf_accuracy, svm_accuracy, gb_accuracy, lstm_accuracy]
```

In [40]:
```
# Visualize Moving Averages
plt.figure(figsize=(12, 6))
plt.plot(data['Close'], label='Close Price', color='blue')
plt.plot(data['15_day_ma'], label='15-day Moving Average', color='red')
plt.plot(data['30_day_ma'], label='30-day Moving Average', color='green')
plt.title('Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



In [41]:
```
# Visualize Stock Prices
plt.figure(figsize=(12, 6))
plt.plot(data['Close'], label='Close Price', color='blue')
plt.title('Stock Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

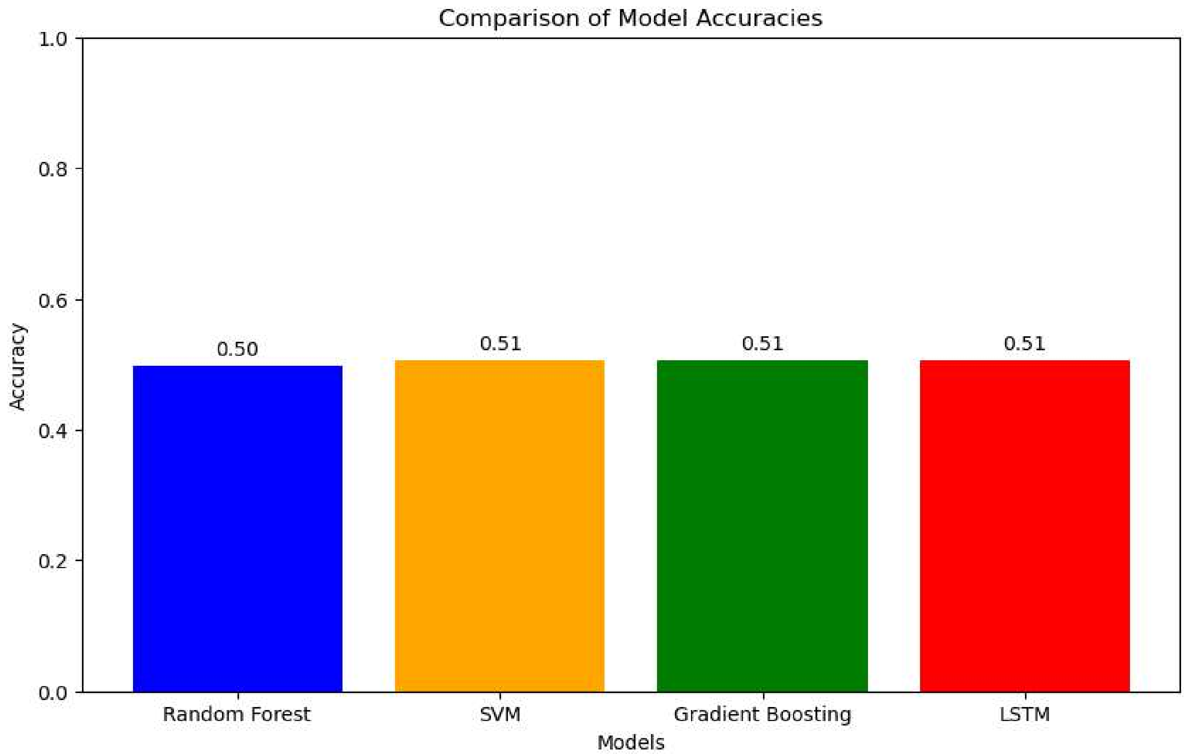


In [21]:
```
best_model = models[np.argmax(accuracies)]
print("Best Model for Stock Market Prediction:", best_model)
```

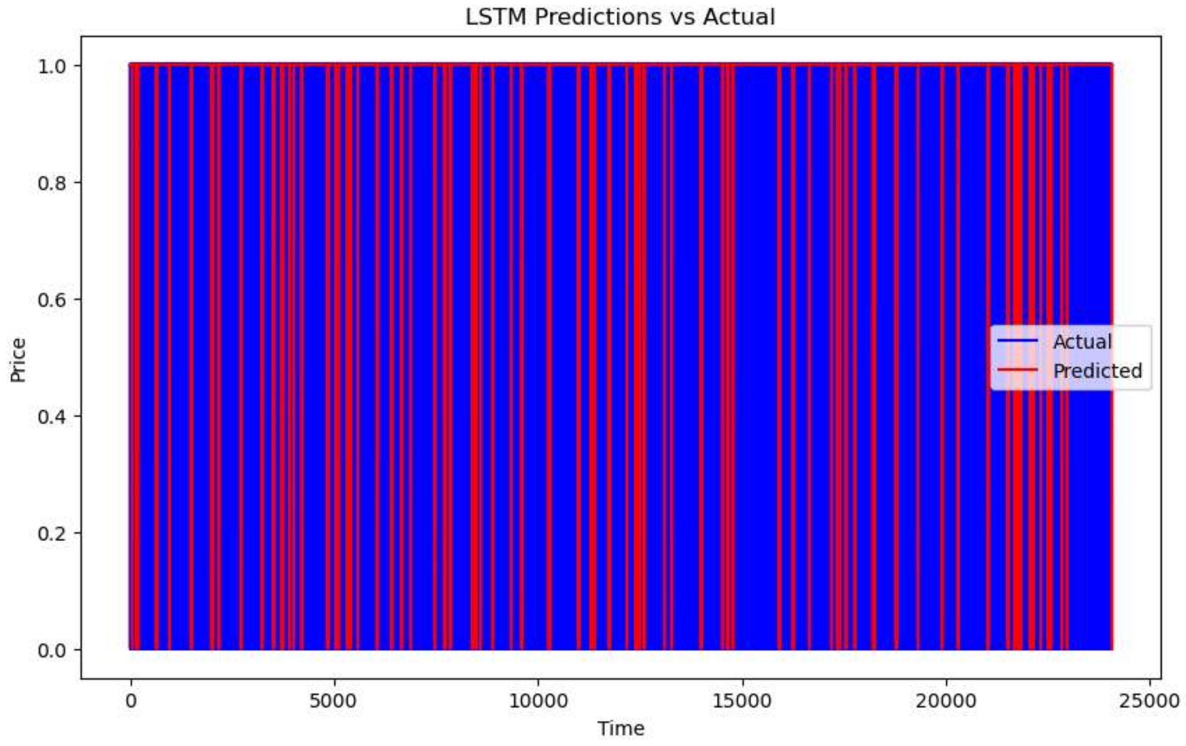
```
Best Model for Stock Market Prediction: LSTM
```

In [24]:
```
# Model Accuracies
models = ['Random Forest', 'SVM', 'Gradient Boosting', 'LSTM']
accuracies = [rf_accuracy, svm_accuracy, gb_accuracy, lstm_accuracy]
```

In [25]:
```
plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color=['blue', 'orange', 'green', 'red'])
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Comparison of Model Accuracies')
plt.ylim(0, 1)
for i, acc in enumerate(accuracies):
    plt.text(i, acc + 0.01, f"{acc:.2f}", ha='center', va='bottom', color='black')
plt.show()
```



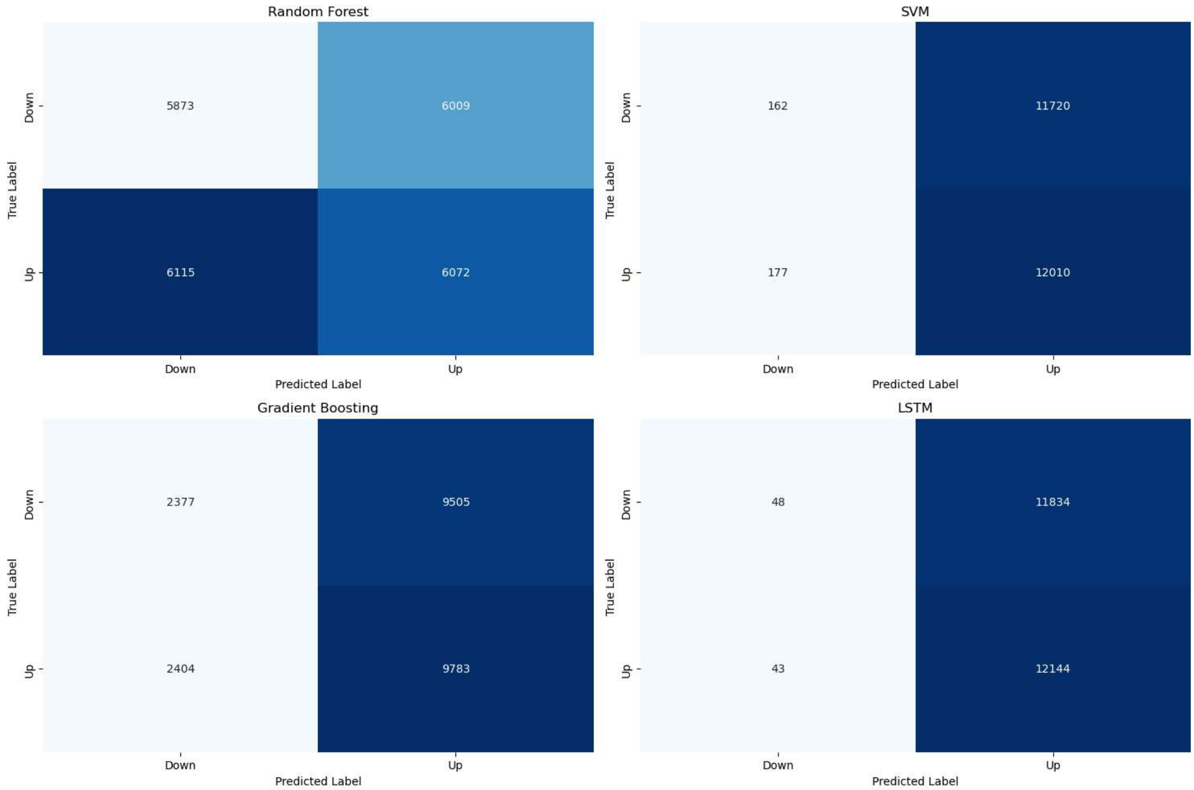
In [26]:
```python
# Visualize LSTM predictions
plt.figure(figsize=(10, 6))
plt.plot(y_test, color='blue', label='Actual')
plt.plot(lstm_pred, color='red', label='Predicted')
plt.title('LSTM Predictions vs Actual')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```



In [27]:
```python
#confusion Matrix
# Visualize the confusion matrix to see the distribution of true positive,
# true negative, false positive, and false negative predictions for each model.
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In [28]:
```python
# Generate confusion matrix for each model
conf_matrices = [confusion_matrix(y_test, pred) for pred in [rf_pred, svm_pred, gb_
```

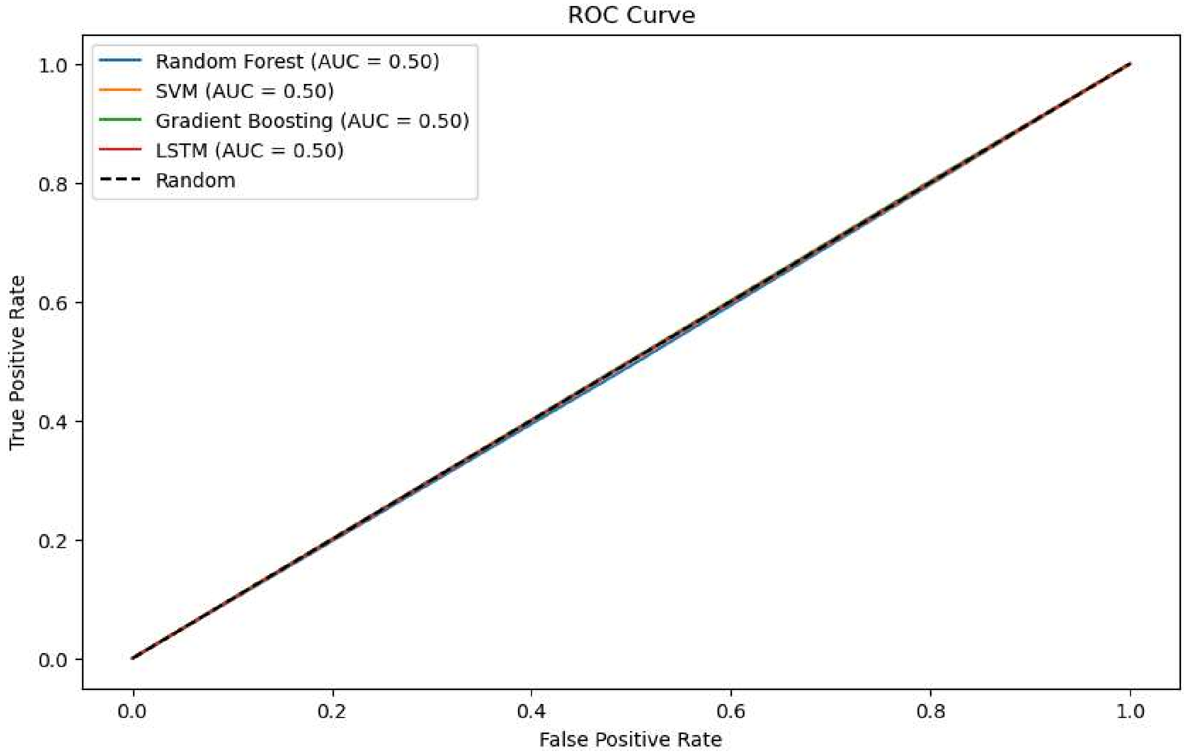
In [29]:
```python
# Plot confusion matrix
plt.figure(figsize=(15, 10))
for i, conf_matrix in enumerate(conf_matrices):
    plt.subplot(2, 2, i + 1)
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Down', 'Up'], yticklabels=['Down', 'Up'])
    plt.title(models[i])
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
plt.tight_layout()
plt.show()
```



In [30]:
```
# ROC Curve
# Plot the Receiver Operating Characteristic (ROC) curve and calculate the Area Und
# score to evaluate the performance of binary classifiers.
from sklearn.metrics import roc_curve, auc
```

In [31]:
```
# Generate ROC curve for each model
roc_curves = [roc_curve(y_test, pred) for pred in [rf_pred, svm_pred, gb_pred, lstm
```

In [32]:
```
# Plot ROC curve
plt.figure(figsize=(10, 6))
for fpr, tpr, model_name in zip([fpr for fpr, _, _ in roc_curves],
                                 [tpr for _, tpr, _ in roc_curves],
                                 models):
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

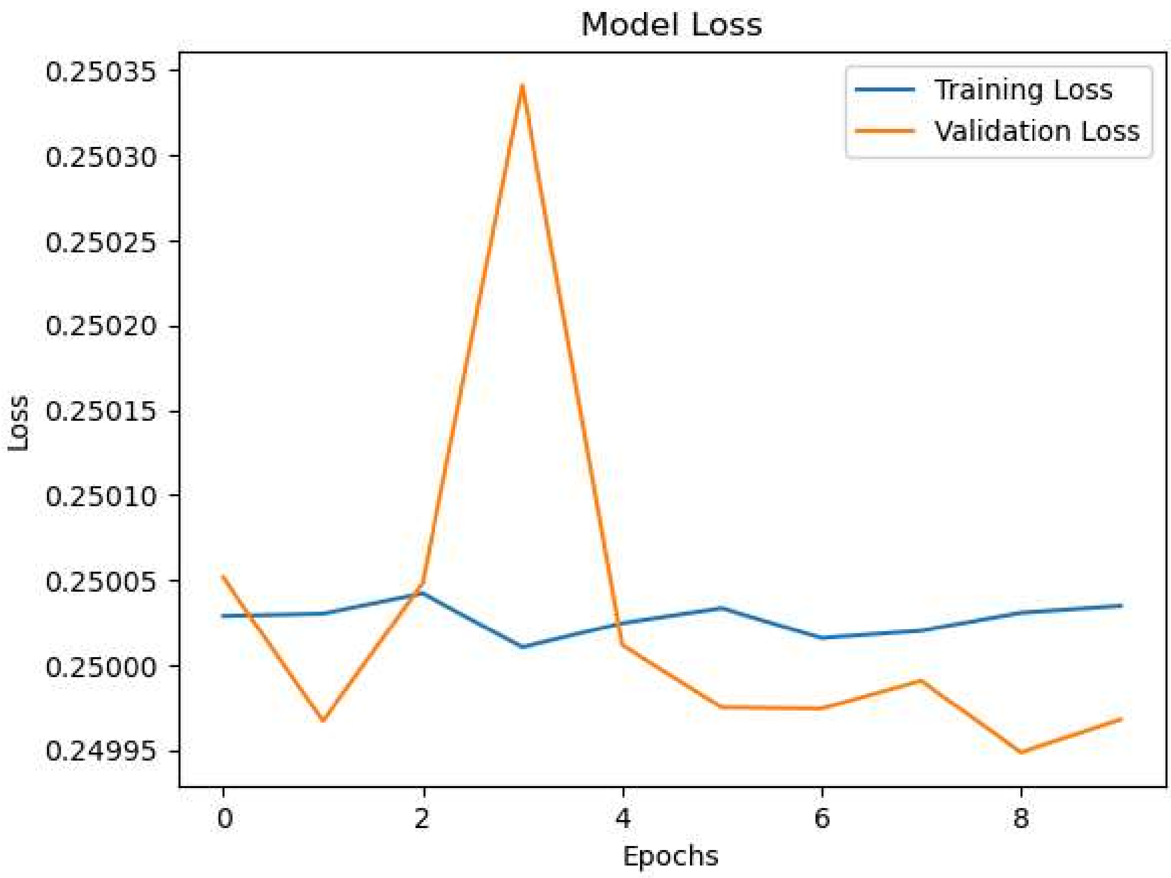


In [33]:
```
# Model Loss Curve (for LSTM)
# Plot the loss curve during the training of the LSTM model to observe the training
```

In [38]:
```
# Train the model
history = model.fit(X_train_lstm, y_train, epochs=10, batch_size=32, validation_dat
```

```
Epoch 1/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 49s 16ms/step - loss: 0.2500 - val_loss: 0.2501
Epoch 2/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 53s 17ms/step - loss: 0.2500 - val_loss: 0.2500
Epoch 3/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 49s 16ms/step - loss: 0.2501 - val_loss: 0.2500
Epoch 4/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 49s 16ms/step - loss: 0.2500 - val_loss: 0.2503
Epoch 5/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 54s 18ms/step - loss: 0.2500 - val_loss: 0.2500
Epoch 6/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 46s 15ms/step - loss: 0.2500 - val_loss: 0.2500
Epoch 7/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 49s 16ms/step - loss: 0.2500 - val_loss: 0.2500
Epoch 8/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 49s 16ms/step - loss: 0.2500 - val_loss: 0.2500
Epoch 9/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 51s 17ms/step - loss: 0.2501 - val_loss: 0.2499
Epoch 10/10
3009/3009 ━━━━━━━━━━━━━━━━━━━━ 55s 18ms/step - loss: 0.2500 - val_loss: 0.2500
```

In [39]:
```
# Plot Model Loss Curve (for LSTM)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [65]:
```
from pptx import Presentation
from pptx.util import Inches
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [66]:
```
# Create a presentation object
prs = Presentation()
```

In [67]:
```
# Slide 1: Title Slide
slide_1 = prs.slides.add_slide(prs.slide_layouts[0])
title = slide_1.shapes.title
subtitle = slide_1.placeholders[1]
title.text = "Stock Market Prediction Analysis"
subtitle.text = "By Kiran Jorwekar"
```

In [68]:
```
# Slide 2: Introduction to Stock Market Prediction
slide_2 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_2.shapes.title
title.text = "Introduction to Stock Market Prediction"
content = slide_2.placeholders[1]
content.text = "Stock market prediction plays a crucial role in financial decision-
               "This presentation explores the use of the NIFTY50_all.csv dataset f
               "stock market movements."
```

In [69]:
```
# Slide 3: Data Overview
slide_3 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_3.shapes.title
title.text = "Data Overview"
content = slide_3.placeholders[1]
content.text = "The NIFTY50_all.csv dataset contains historical stock market data,
               "opening and closing prices, trading volume, and other relevant feat
               "Let's begin by exploring the dataset."
```

In [70]:
```
# Slide 4: Data Preprocessing
slide_4 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_4.shapes.title
title.text = "Data Preprocessing"
content = slide_4.placeholders[1]
content.text = "Before building predictive models, it's important to preprocess the
               "This includes handling missing values, outliers, and feature engine
```

In [71]:
```
# Slide 5: Data Visualization
slide_5 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_5.shapes.title
title.text = "Data Visualization"
content = slide_5.placeholders[1]
content.text = "Visualizing the data can provide insights into stock price trends,
               "distribution of features, and other patterns that may impact predic
```

In [72]:
```
# Slide 6: Model Selection
slide_6 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_6.shapes.title
title.text = "Model Selection"
content = slide_6.placeholders[1]
content.text = "Several algorithms can be considered for stock market prediction. "
               "In this analysis, we will focus on the Long Short-Term Memory (LSTM
               "algorithm due to its ability to capture long-term dependencies."
```

In [73]:
```
# Slide 7: Model Training and Evaluation
slide_7 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_7.shapes.title
title.text = "Model Training and Evaluation"
content = slide_7.placeholders[1]
content.text = "The LSTM model will be trained using 15 or 30 days of historical da
               "Evaluation metrics such as accuracy, precision, and recall will be
               "to assess the performance of the model."
```

In [74]:
```
# Slide 8: Results and Analysis
slide_8 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_8.shapes.title
title.text = "Results and Analysis"
content = slide_8.placeholders[1]
content.text = "The analysis will present accuracy details for each model and compa
               "the performance of LSTM with other algorithms. Suggestions on which
               "model is best suited for stock market prediction will also be provi
```

In [75]:
```
# Slide 9: Conclusion
slide_9 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_9.shapes.title
title.text = "Conclusion"
content = slide_9.placeholders[1]
content.text = "In conclusion, stock market prediction using the NIFTY50_all.csv da
               "involves preprocessing the data, selecting appropriate models, trai
               "and evaluating them, and analyzing the results to make informed dec
               "regarding future market movements."
```

In [76]:
```
# Slide 10: Code Snippets
slide_10 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_10.shapes.title
title.text = "Code Snippets"
content = slide_10.placeholders[1]
content.text = "Code snippets from the analysis notebook will be included to showca
               "data preprocessing, model training, evaluation, and visualization s
```

In [77]:
```
# Slide 11: References
slide_11 = prs.slides.add_slide(prs.slide_layouts[1])
title = slide_11.shapes.title
title.text = "References"
content = slide_11.placeholders[1]
content.text = "Any references or resources used in the analysis will be listed her
```

In [78]:
```
# Save the presentation
prs.save("stock_market_analysis_presentation.pptx")
```

In [ ]:
```

```