Report: Stock Sentiment Analysis Using Machine Learning Techniques

Title: Stock Sentiment Analysis Using Machine Learning Techniques (#FC24OPS3)

Overview

The project aims to predict stock price movements based on sentiment analysis of textual data from news articles, social media, and other financial news sources. By analyzing the sentiment expressed in these texts, the model seeks to uncover insights into investor and market sentiment, valuable for making informed trading decisions.

Objectives

1. Data Collection: Gather a large dataset of textual data related to stocks, including news articles, social media posts, earnings reports, and analyst reports.

2. Data Preprocessing: Clean the textual data by removing noise, tokenizing text, and applying stemming and lemmatization to standardize text representations.

3. Data Labeling: Label the textual data with corresponding stock price movements (e.g., increase, decrease, no change) over a specified time horizon for supervised learning.

4. Feature Extraction: Extract features from the textual data, such as word frequencies, sentiment scores, and topic modeling representations, to represent the text for machine learning algorithms.

5. Model Training and Evaluation: Train and evaluate machine learning models, including logistic regression, SVM, random forests, and neural networks, to predict stock price movements based on textual sentiment.

6. Model Evaluation: Use metrics such as accuracy, precision, recall, F1-score, and ROC curve analysis to assess the model's performance.

Scope

- Focus: Sentiment analysis for a specific set of stocks, chosen based on data availability and relevance.

- Data Sources: Financial news websites, social media platforms (e.g., Twitter, StockTwits), and regulatory filings (e.g., SEC filings).

Code Explanation

Data Loading and Preprocessing

```
import pandas as pd
import numpy as np
from google.colab import drive

drive.mount('/content/drive')
file_path = '/content/drive/My Drive/Colab/Data.csv'
df = pd.read_csv(file_path, encoding="ISO-8859-1")
```

- Mounting Google Drive: Access the dataset stored in Google Drive.

- Loading Data: Read the CSV file containing the dataset.

Data Splitting

```
train = df[df['Date'] < '20150101']
test = df[df['Date'] > '20141231']
```

- Training and Testing Split: Split the data into training and testing sets based on the date.

Text Preprocessing

```
data = train.iloc[:, 2:27]
data.replace("[^a-zA-Z]", " ", regex=True, inplace=True)
list1 = [i for i in range(25)]
new_Index = [str(i) for i in list1]
data.columns = new_Index
for index in new_Index:
    data[index] = data[index].str.lower()
```

- Cleaning Data: Remove punctuations and convert text to lowercase.

- Renaming Columns: Rename columns for easier access.

Feature Extraction

```
headlines = []
for row in range(0, len(data.index)):
    headlines.append(' '.join(str(x) for x in data.iloc[row, 0:25]))

from sklearn.feature_extraction.text import CountVectorizer
countvector = CountVectorizer(ngram_range=(2,2))
traindataset = countvector.fit_transform(headlines)
```

- Combining Headlines: Combine all headlines for each row into a single string.

- Vectorization: Use CountVectorizer to convert text data into numerical format using bigrams.

Model Training

```
from sklearn.ensemble import RandomForestClassifier
randomclassifier = RandomForestClassifier(n_estimators=200,
criterion='entropy')
randomclassifier.fit(traindataset, train['Label'])
```

- RandomForest Classifier: Train a RandomForest model to predict stock movements.

Model Evaluation

```
import matplotlib.pyplot as plt
from sklearn import metrics
import itertools
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion
matrix', cmap=plt.cm.Blues):
    # Plotting code

test_transform = []
for row in range(0, len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row, 2:27]))
test_dataset = countvector.transform(test_transform)
predictions = randomclassifier.predict(test_dataset)

matrix = confusion_matrix(test['Label'], predictions)
plot_confusion_matrix(matrix, classes=['Down', 'Up'])
score = accuracy_score(test['Label'], predictions)
```

```
report = classification_report(test['Label'], predictions)

print(matrix)
print(round(score*100, 2))
print(report)
```

- Prediction and Evaluation: Predict stock movements using the test data and evaluate the model using confusion matrix, accuracy score, and classification report.

Alternative Approach: TF-IDF Vectorizer and Naive Bayes

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfvector = TfidfVectorizer(ngram_range=(2,2))
traindataset = tfidfvector.fit_transform(headlines)

randomclassifier = RandomForestClassifier(n_estimators=200,
criterion='entropy')
randomclassifier.fit(traindataset, train['Label'])

# Repeat evaluation steps for TF-IDF and RandomForest

from sklearn.naive_bayes import MultinomialNB
naive = MultinomialNB()
naive.fit(traindataset, train['Label'])

# Repeat evaluation steps for Naive Bayes
```

- TF-IDF Vectorizer: Use TF-IDF for feature extraction and evaluate with RandomForest.

- Naive Bayes Classifier: Train and evaluate using Naive Bayes classifier.

Conclusion

The report demonstrates the development of a sentiment analysis model using machine learning techniques to predict stock price movements. The project includes data collection, preprocessing, feature extraction, model training, and evaluation.

Performance Metrics

- Accuracy Score: Measures the overall correctness of the model.

- Confusion Matrix: Evaluates the performance in distinguishing between different classes.

- Classification Report: Provides precision, recall, and F1-score.

- Sharpe Ratio, Maximum Drawdowns, Number of Trades, Win Ratio: Additional metrics to judge trading strategy performance based on sentiment analysis.

Future Work

The project acknowledges challenges such as language ambiguity and data quality. Future work may involve improving NLP techniques and model recalibration to adapt to changing market conditions.