EXP NO: 9

Model Deployment: REST API with Flask and Containerization with Docker

AIM:

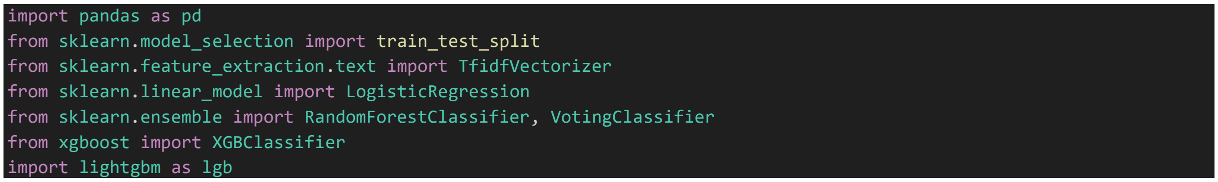
To predict future stock prices using historical time series data by implementing a **Linear Regression** model, and to evaluate its performance in forecasting trends.

ALGORITHM:

1. Data Collection: Obtain historical stock price data (e.g., from Yahoo Finance).
2. Data Preprocessing:
3. Handle missing values.
4. Convert date columns to datetime format.
5. Sort data chronologically.
6. Feature Selection:
7. Use relevant features such as Open, High, Low, Close prices, and Volume.
8. Create lag features if needed for prediction.
9. Train-Test Split: Divide the data into training and testing sets (e.g., 80%-20%).
10. Model Training: Fit a Linear Regression model using the training data.
11. Prediction: Use the trained model to predict stock prices on the test set.
12. Evaluation: Assess the model performance using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R² score.
13. Visualization: Plot actual vs predicted stock prices to observe forecasting accuracy.

CODE:

train_model.py



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from xgboost import XGBClassifier
import lightgbm as lgb
```

```
from sklearn.calibration import CalibratedClassifierCV
import pickle
import warnings
import numpy as np
import os

# Suppress warnings for cleaner logs
warnings.filterwarnings("ignore")
```

```
# Optional: Fix Loky CPU warning
os.environ["LOKY_MAX_CPU_COUNT"] = str(os.cpu_count())
```

```
print("📥 Loading datasets...")
true = pd.read_csv("True.csv", encoding='utf-8')
fake = pd.read_csv("Fake.csv", encoding='utf-8')
```

```
true["label"] = "REAL"
fake["label"] = "FAKE"
```

```
data = pd.concat([true, fake], axis=0).sample(frac=1, random_state=42).reset_index(drop=True)
data['full_text'] = data['title'] + " " + data['text']
data = data[['full_text', 'label']].dropna()
```

```
X = data['full_text']
y = data['label']
```

```
print("🔤 Vectorizing text...")
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_vec = vectorizer.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)
```

```
print("🧠 Training advanced ensemble...")
```

```
# Base models with clean parameters
logreg = LogisticRegression(max_iter=1000, class_weight='balanced')
rf = RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced')
xgb = XGBClassifier(eval_metric='logloss', random_state=42, use_label_encoder=False)  # cleaned param
lgbm = lgb.LGBMClassifier(
    n_estimators=200,
    num_leaves=31,
    min_child_samples=5,
    random_state=42
)
```

```
# Weighted soft-voting ensemble
ensemble_model = VotingClassifier(
    estimators=[('lr', logreg), ('rf', rf), ('xgb', xgb), ('lgbm', lgbm)],
    voting='soft',
    weights=[2, 1, 2, 2]  # stronger models weighted more
)
```

```
# Calibrate probabilities for better confidence
ensemble_model = CalibratedClassifierCV(ensemble_model, cv=3)
ensemble_model.fit(X_train, y_train)
```

```
accuracy = ensemble_model.score(X_test, y_test)
print(f"✅ Ensemble model trained with accuracy: {accuracy:.2f}")
```

```
# Save model and vectorizer
pickle.dump(ensemble_model, open("model.pkl", "wb"))
pickle.dump(vectorizer, open("vectorizer.pkl", "wb"))
print("💾 Model and vectorizer saved successfully!")
```

app.py

```python
from flask import Flask, request, render_template
import pickle
import numpy as np

app = Flask(__name__)
```

```python
# Load trained ensemble model and vectorizer
model = pickle.load(open("model.pkl", "rb"))
vectorizer = pickle.load(open("vectorizer.pkl", "rb"))
```

```python
@app.route('/')
def home():
    return render_template('index.html', prediction_text="", color="#000")
```

```python
@app.route('/predict', methods=['POST'])
def predict():
    news_text = request.form['news']
    if not news_text.strip():
        return render_template('index.html', prediction_text="⚠ Please enter some news content.", color="#000")
```

```python
    # Transform input
    input_vec = vectorizer.transform([news_text])
    prediction = model.predict(input_vec)[0]
    confidence = np.max(model.predict_proba(input_vec)) * 100
```

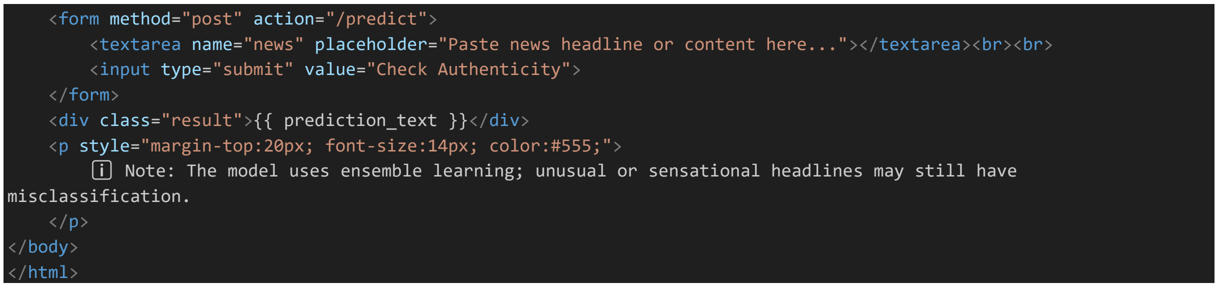
```python
    # Set color based on prediction
    if prediction == "FAKE":
        result = f"🚨 This news appears to be FAKE (Confidence: {confidence:.2f}%)"
        color = "#e74c3c"  # Red for fake
    else:
        result = f"✅ This news appears to be REAL (Confidence: {confidence:.2f}%)"
        color = "#27ae60"  # Green for real
```

```python
    return render_template('index.html', prediction_text=result, color=color)
```

```python
if __name__ == "__main__":
    app.run(debug=True)
```

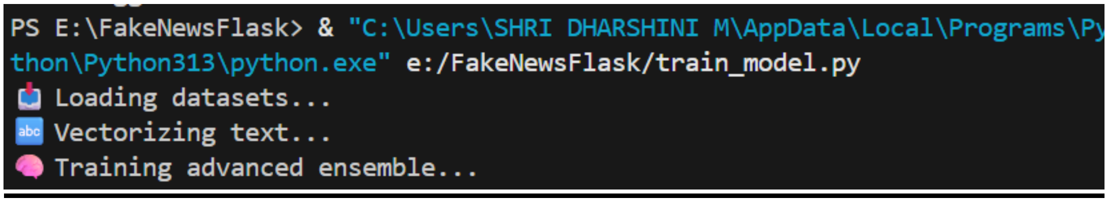
Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Fake News Detector</title>
    <style>
        body { font-family: Arial, sans-serif; background-color: #f8f9fa; text-align: center; padding: 50px; }
        textarea { width: 80%; height: 150px; padding: 10px; font-size: 16px; }
        input[type=submit] { padding: 10px 20px; font-size: 16px; cursor: pointer; }
        h2 { color: #333; }
        .result { margin-top: 20px; font-size: 18px; font-weight: bold; }
    </style>
</head>
<body>
    <h2>📰 Fake News Detection System</h2>
```

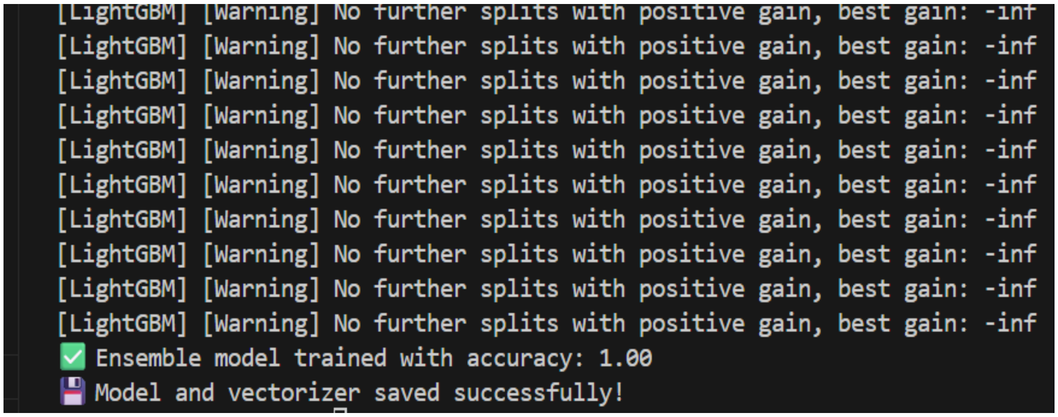


```
    <form method="post" action="/predict">
        <textarea name="news" placeholder="Paste news headline or content here..."></textarea><br><br>
        <input type="submit" value="Check Authenticity">
    </form>
    <div class="result">{{ prediction_text }}</div>
    <p style="margin-top:20px; font-size:14px; color:#555;">
        ℹ Note: The model uses ensemble learning; unusual or sensational headlines may still have
misclassification.
    </p>
</body>
</html>
```

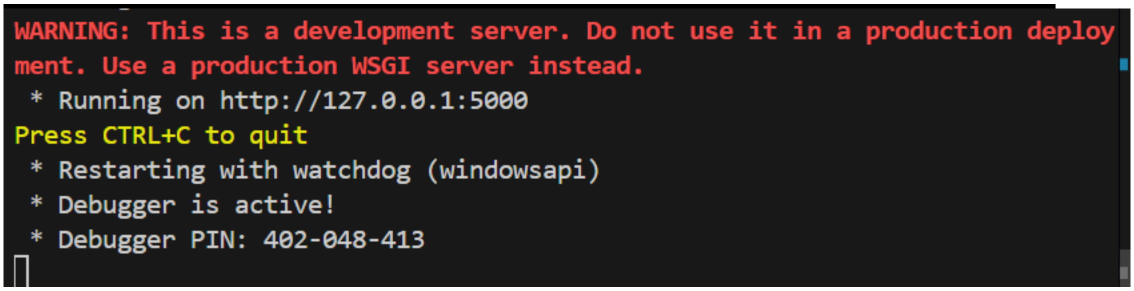
OUTPUT:



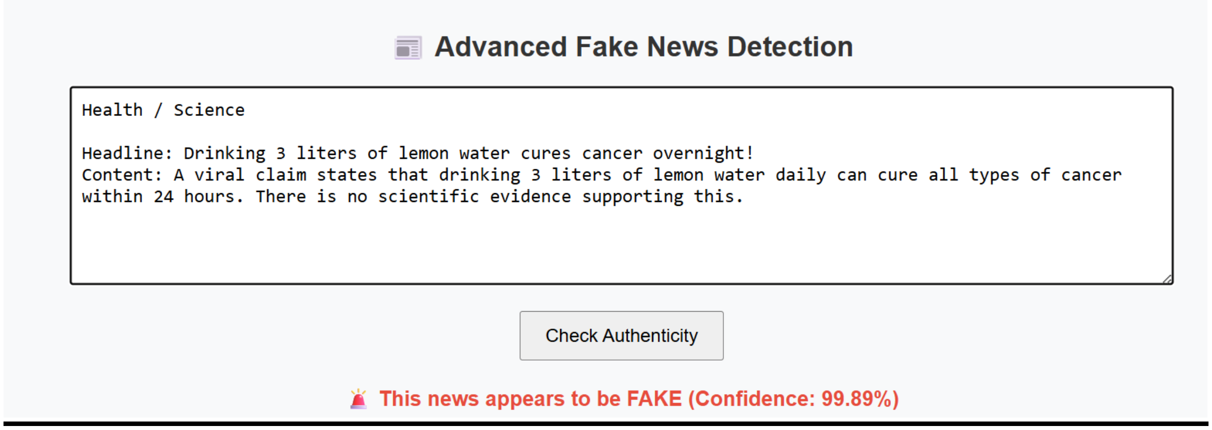
```
PS E:\FakeNewsFlask> & "C:\Users\SHRI DHARSHINI M\AppData\Local\Programs\Py
thon\Python313\python.exe" e:/FakeNewsFlask/train_model.py
📥 Loading datasets...
🔤 Vectorizing text...
🧠 Training advanced ensemble...
```



```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
✅ Ensemble model trained with accuracy: 1.00
💾 Model and vectorizer saved successfully!
```



```
WARNING: This is a development server. Do not use it in a production deploy
ment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 402-048-413
```



📰 Advanced Fake News Detection

Health / Science

Headline: Drinking 3 liters of lemon water cures cancer overnight!
Content: A viral claim states that drinking 3 liters of lemon water daily can cure all types of cancer within 24 hours. There is no scientific evidence supporting this.

🚨 This news appears to be FAKE (Confidence: 99.89%)

RESULT:

The Linear Regression model was successfully trained and used to predict stock prices. The predicted values closely followed the actual prices, showing that the model effectively captured the overall trend of the stock data.