RAPID Internship Project Code

Flask App Main Code : Includes ANN, GBM/XgBoost, LSTM, Random Forest models training and testing

```
from flask import Flask, request, jsonify, render_template
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
import io
import base64

app = Flask(__name__)

# Function to load data
def load_data(file):
    data = pd.read_csv(file)
    data['Date'] = pd.to_datetime(data['Date'], format='%m/%d/%y')
    data.set_index('Date', inplace=True)
    data = data.fillna(method='ffill')
    return data

def prepare_ann_data(data, seq_length):
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(data[['Close']])

    X, y = [], []
    for i in range(seq_length, len(scaled_data)):
        X.append(scaled_data[i-seq_length:i, 0])
        y.append(scaled_data[i, 0])
    X, y = np.array(X), np.array(y)

    return X, y, scaler
# Function to prepare data for ANN and LSTM
def prepare_ann_lstm_data(data, seq_length):
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(data[['Close']])

    X, y = [], []
    for i in range(seq_length, len(scaled_data)):
        X.append(scaled_data[i-seq_length:i, 0])
        y.append(scaled_data[i, 0])
    X, y = np.array(X), np.array(y)

    return X, y, scaler

def prepare_rf_data(data, seq_length):
    X, y = [], []
    for i in range(seq_length, len(data)):
        X.append(data['Close'].iloc[i-seq_length:i].values)
        y.append(data['Close'].iloc[i])
    return np.array(X), np.array(y)

def prepare_gbm_data(data, seq_length):
    X, y = [], []
    for i in range(seq_length, len(data)):
        X.append(data['Close'].iloc[i-seq_length:i].values)
        y.append(data['Close'].iloc[i])
    return np.array(X), np.array(y)
# Function to prepare data for GBM and RFR
def prepare_gbm_rf_data(data, seq_length):
```

```
    X, y = [], []
    for i in range(seq_length, len(data)):
        X.append(data['Close'].iloc[i-seq_length:i].values)
        y.append(data['Close'].iloc[i])
    return np.array(X), np.array(y)

# Function to build ANN model
def build_ann_model(input_dim):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(50, input_dim=input_dim, activation='relu'),
        tf.keras.layers.Dense(25, activation='relu'),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Function to build LSTM model
def build_lstm_model(seq_length):
    model = tf.keras.Sequential([
        tf.keras.layers.LSTM(50, return_sequences=True, input_shape=(seq_length,
1)),
        tf.keras.layers.LSTM(50, return_sequences=False),
        tf.keras.layers.Dense(25),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Function to build GRU model
def build_gru_model(seq_length):
    model = tf.keras.Sequential([
        tf.keras.layers.GRU(50, return_sequences=True, input_shape=(seq_length,
1)),
        tf.keras.layers.GRU(50, return_sequences=False),
        tf.keras.layers.Dense(25),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Function to plot results
def plot_results(train, test, predictions, model_name, seq_length):
    plt.figure(figsize=(12, 6))
    plt.plot(train.index, train['Close'], label='Training Data')
    plt.plot(test.index, test['Close'], label='Actual Prices', color='blue')
    plt.plot(test.index[seq_length:], predictions, label=f'{model_name}
Predictions', color='red')
    plt.legend()
    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()
    plt.close()
    return plot_url

@app.route('/prediction')
def index():
    return render_template('prediction.html')

@app.route('/')
def homepage():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    crypto = request.form['crypto']
    model_choice = request.form['model']
    file = request.files['csvFile']
```

```
    seq_length = int(request.form['seq_length'])

    # Load data based on cryptocurrency choice
    data = load_data(file)

    train_size = int(len(data) * 0.8)
    train, test = data.iloc[:train_size], data.iloc[train_size:]

    if model_choice == 'ANN':
        X, y, scaler = prepare_ann_data(data, seq_length)
        X_train, y_train = X[:train_size-seq_length], y[:train_size-seq_length]
        X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]

        ann_model = build_ann_model(seq_length)
        ann_model.fit(X_train, y_train, batch_size=1, epochs=5)

        ann_pred_scaled = ann_model.predict(X_test)
        ann_pred = scaler.inverse_transform(ann_pred_scaled).flatten()

        test_true = data['Close'].values[train_size:]

        if len(ann_pred) != len(test_true):
            print("Warning: Lengths of ann_pred and test_true do not match!")
            min_len = min(len(ann_pred), len(test_true))
            ann_pred = ann_pred[:min_len]
            test_true = test_true[:min_len]

        plt.figure(figsize=(12, 6))
        plt.plot(train.index, train['Close'], label='Training Data')
        plt.plot(test.index, test['Close'], label='Actual Prices', color='blue')
        plt.plot(test.index, ann_pred, label='ANN Predictions', color='red')
        plt.legend()
        img = io.BytesIO()
        plt.savefig(img, format='png')
        img.seek(0)
        plot_url = base64.b64encode(img.getvalue()).decode()
        plt.close()


        def calculate_mape(y_true, y_pred):
            return np.mean(np.abs((y_true - y_pred) / y_true))

        mse = np.sqrt(mean_squared_error(test_true, ann_pred))
        mape = calculate_mape(test_true, ann_pred)

        return jsonify({'mse': mse, 'mape': mape, 'plot_url': plot_url})

    elif model_choice == 'LSTM':
        X, y, scaler = prepare_ann_lstm_data(data, seq_length)
        X_train, y_train = X[:train_size], y[:train_size]  # Use full training set
        X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]  #
Use full test set

        X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
        X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

        lstm_model = build_lstm_model(seq_length)
        lstm_model.fit(X_train, y_train, batch_size=1, epochs=1)

        lstm_pred = lstm_model.predict(X_test)
        lstm_pred = scaler.inverse_transform(lstm_pred)
        def plot_results(train, test, predictions, model_name):
            plt.figure(figsize=(12, 6))
            plt.plot(train.index, train['Close'], label='Training Data')
            plt.plot(test.index, test['Close'], label='Actual Prices',
color='blue')

            # Adjust predictions to align with test data indices
```

```
            predictions_index = test.index[:len(predictions)]
            plt.plot(predictions_index, predictions, label=f'{model_name}
Predictions', color='red')

            plt.legend()
            img = io.BytesIO()
            plt.savefig(img, format='png')
            img.seek(0)
            plot_url = base64.b64encode(img.getvalue()).decode()
            plt.close()
            return plot_url
        def compute_accuracy(y_true, y_pred):
            rmse = np.sqrt(mean_squared_error(y_true, y_pred))
            mape = mean_absolute_percentage_error(y_true, y_pred)
            return rmse, mape
                # Scale back the y_test for accuracy computation
        y_test_scaled = scaler.inverse_transform(y_test.reshape(-1, 1))
        rmse, mape = compute_accuracy(y_test_scaled, lstm_pred)
        plot_url= plot_results(train, test, lstm_pred, 'LSTM')
        return jsonify({'mse': rmse, 'mape': mape, 'plot_url': plot_url})

    elif model_choice == 'LSTM-GRU':
        X, y, scaler = prepare_ann_lstm_data(data, seq_length)
        X_train, y_train = X[:train_size], y[:train_size]  # Use full training set
        X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]  #
Use full test set
        X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
        X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
        lstm_model = build_lstm_model(seq_length)
        lstm_model.fit(X_train, y_train, batch_size=1, epochs=1)
        gru_model = build_gru_model(seq_length)
        gru_model.fit(X_train, y_train, batch_size=1, epochs=1)
        lstm_pred = lstm_model.predict(X_test)
        gru_pred = gru_model.predict(X_test)
        lstm_pred = scaler.inverse_transform(lstm_pred)
        gru_pred = scaler.inverse_transform(gru_pred)
        alpha = 0.5  # Weight parameter for exponential formula
        ensemble_pred = alpha * lstm_pred + (1 - alpha) * gru_pred
        y_test_scaled = scaler.inverse_transform(y_test.reshape(-1, 1))
        def compute_accuracy(y_true, y_pred):
            rmse = np.sqrt(mean_squared_error(y_true, y_pred))
            mape = mean_absolute_percentage_error(y_true, y_pred)
            return rmse, mape
        def plot_results(train, test, predictions, model_name):
            plt.figure(figsize=(12, 6))
            plt.plot(train.index, train['Close'], label='Training Data')
            plt.plot(test.index, test['Close'], label='Actual Prices',
color='blue')

            # Adjust predictions to align with test data indices
            predictions_index = test.index[:len(predictions)]
            plt.plot(predictions_index, predictions, label=f'{model_name}
Predictions', color='red')

            plt.legend()
            img = io.BytesIO()
            plt.savefig(img, format='png')
            img.seek(0)
            plot_url = base64.b64encode(img.getvalue()).decode()
            plt.close()
            return plot_url
        rmse, mape = compute_accuracy(y_test_scaled, ensemble_pred)
        plot_url= plot_results(train, test, ensemble_pred, 'Ensemble Without
Sentiment Predictions')
        return jsonify({'mse': rmse, 'mape': mape, 'plot_url': plot_url})

    elif model_choice == 'GBM':
        X, y = prepare_gbm_data(data, seq_length)
```

```
        X_train, y_train = X[:train_size-seq_length], y[:train_size-seq_length]
        X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]

        gbm_model = XGBRegressor(n_estimators=100, learning_rate=0.1)
        gbm_model.fit(X_train, y_train)
        gbm_pred = gbm_model.predict(X_test)

        mse = np.sqrt(mean_squared_error(y_test, gbm_pred))
        mape = mean_absolute_percentage_error(y_test, gbm_pred)

        plt.figure(figsize=(12, 6))
        plt.plot(data.index, data['Close'], label='Actual Prices')
        plt.plot(data.index[train_size:], gbm_pred, label='GBM Predictions',
color='red')
        plt.legend()
        img = io.BytesIO()
        plt.savefig(img, format='png')
        img.seek(0)
        plot_url = base64.b64encode(img.getvalue()).decode()
        plt.close()
        return jsonify({'mse': mse, 'mape': mape, 'plot_url': plot_url})

    elif model_choice == 'RFR':
        X, y = prepare_rf_data(data, seq_length)
        X_train, y_train = X[:train_size-seq_length], y[:train_size-seq_length]
        X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]

        rf_model = RandomForestRegressor(n_estimators=100)
        rf_model.fit(X_train, y_train)
        rf_pred = rf_model.predict(X_test)

        rf_mse = mean_squared_error(y_test, rf_pred)
        rf_mape = mean_absolute_percentage_error(y_test, rf_pred)
        plt.figure(figsize=(12, 6))
        plt.plot(data.index, data['Close'], label='Actual Prices')
        plt.plot(data.index[train_size:], rf_pred, label='Random Forest
Predictions', color='red')
        plt.legend()
        img = io.BytesIO()
        plt.savefig(img, format='png')
        img.seek(0)
        plot_url = base64.b64encode(img.getvalue()).decode()
        plt.close()
        return jsonify({'mse': rf_mse, 'mape': rf_mape, 'plot_url': plot_url})

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

Code for Twitter database Sentiment Analysis

```
import pandas as pd
import re
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from transformers import pipeline
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import nltk
from tqdm import tqdm  # Import tqdm for progress bar

# Download NLTK data files
nltk.download('wordnet')
nltk.download('stopwords')

# Load the CSV file
file_path = 'data/tweets_test.csv'
```

```
data = pd.read_csv(file_path)

# Initialize VADER sentiment analyzer and BERT model
analyzer = SentimentIntensityAnalyzer()
bert_pipeline = pipeline("sentiment-analysis")

# Ensure all values in 'full_text' are strings
data['full_text'] = data['full_text'].fillna('').astype(str)

# Function to clean text
def clean_text(text):
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)  #
Remove URLs
    text = re.sub(r'\@\w+|\#', '', text)  # Remove mentions and hashtags
    text = re.sub(r'[^A-Za-z\s]', '', text)  # Remove special characters and
numbers
    text = text.lower()  # Convert to lowercase
    text = ' '.join([word for word in text.split() if word not in
stopwords.words('english')])  # Remove stop words
    lemmatizer = WordNetLemmatizer()
    text = ' '.join([lemmatizer.lemmatize(word) for word in text.split()])  #
Lemmatization
    return text

# Function to get sentiment compound score
def get_vader_sentiment(text):
    cleaned_text = clean_text(text)
    sentiment = analyzer.polarity_scores(cleaned_text)
    return sentiment['compound']

# Function to get BERT sentiment score
def get_bert_sentiment(text):
    cleaned_text = clean_text(text)
    sentiment = bert_pipeline(cleaned_text)
    return sentiment[0]['score'] if sentiment[0]['label'] == 'POSITIVE' else -
sentiment[0]['score']

# Apply sentiment analysis to the 'full_text' column with progress bar
tqdm.pandas()  # Initialize tqdm with pandas
data['vader_sentiment'] = data['full_text'].progress_apply(get_vader_sentiment)
data['bert_sentiment'] = data['full_text'].progress_apply(get_bert_sentiment)

# Combine the sentiment scores (simple average in this case)
data['sentiment_compound'] = (data['vader_sentiment'] + data['bert_sentiment']) / 2

# Save the result to a new CSV file
output_file_path = 'data/sentiment_analysis_output_advanced.csv'
data.to_csv(output_file_path, index=False)

print(f"Sentiment analysis results saved to {output_file_path}")
```

Code for Training and Testing LSTM-GRU Ensemble model which includes Sentiment scores

Training:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

# Function to load data
def load_data(price_file_path, sentiment_file_path):
    price_data = pd.read_csv(price_file_path)
    sentiment_data = pd.read_csv(sentiment_file_path)
```

```
    price_data['Date'] = pd.to_datetime(price_data['Date'], format='%m/%d/%y')
    price_data.set_index('Date', inplace=True)
    price_data = price_data.fillna(method='ffill')

    sentiment_data['Date'] = pd.to_datetime(sentiment_data['created_at'],
format='%Y-%m-%d')
    sentiment_data.set_index('Date', inplace=True)

    # Combine price data and sentiment scores using outer join
    data = price_data.join(sentiment_data[['sentiment_compound']], how='outer')

    # Fill missing sentiment scores with default value (e.g., 0)
    data['sentiment_compound'].fillna(0, inplace=True)

    # Fill any remaining NaN values in price data
    data.fillna(method='ffill', inplace=True)
    data.fillna(method='bfill', inplace=True)

    return data

# Function to prepare data for LSTM/GRU
def prepare_lstm_data(data, seq_length, use_sentiment=True):
    if not use_sentiment:
        data = data.drop(columns=['sentiment_compound'])

    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(data)

    X, y = [], []
    for i in range(seq_length, len(scaled_data)):
        X.append(scaled_data[i-seq_length:i])
        y.append(scaled_data[i, 0])  # Assuming 'Close' is the first column
    X, y = np.array(X), np.array(y)

    return X, y, scaler

# Function to build LSTM model
def build_lstm_model(seq_length, num_features):
    model = tf.keras.Sequential([
        tf.keras.layers.LSTM(50, return_sequences=True, input_shape=(seq_length,
num_features)),
        tf.keras.layers.LSTM(50, return_sequences=False),
        tf.keras.layers.Dense(25),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Function to build GRU model
def build_gru_model(seq_length, num_features):
    model = tf.keras.Sequential([
        tf.keras.layers.GRU(50, return_sequences=True, input_shape=(seq_length,
num_features)),
        tf.keras.layers.GRU(50, return_sequences=False),
        tf.keras.layers.Dense(25),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Main function
def main():
    # Load data
    data = load_data('data/BTC-USD.csv',
'data/sentiment_analysis_output_advanced.csv')
    train_size = int(len(data) * 0.8)
    train = data.iloc[:train_size]
```

```
    seq_length = 60

    # With sentiment data
    print("\nTraining models with sentiment data:")
    X, y, scaler = prepare_lstm_data(data, seq_length, use_sentiment=True)
    X_train, y_train = X[:train_size], y[:train_size]

    num_features = X_train.shape[2]
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], num_features))

    # Build and train LSTM model
    lstm_model = build_lstm_model(seq_length, num_features)
    lstm_model.fit(X_train, y_train, batch_size=1, epochs=50)
    lstm_model.save('lstm_model.h5')

    # Build and train GRU model
    gru_model = build_gru_model(seq_length, num_features)
    gru_model.fit(X_train, y_train, batch_size=1, epochs=50)
    gru_model.save('gru_model.h5')

    # Save the scaler for later use
    np.save('scaler.npy', scaler.scale_)
    np.save('scaler_min.npy', scaler.min_)

if __name__ == "__main__":
    main()
```

Testing/Predicting:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

# Function to load data
def load_data(price_file_path, sentiment_file_path):
    price_data = pd.read_csv(price_file_path)
    sentiment_data = pd.read_csv(sentiment_file_path)

    price_data['Date'] = pd.to_datetime(price_data['Date'], format='%m/%d/%y')
    price_data.set_index('Date', inplace=True)
    price_data = price_data.fillna(method='ffill')

    sentiment_data['Date'] = pd.to_datetime(sentiment_data['created_at'],
format='%Y-%m-%d')
    sentiment_data.set_index('Date', inplace=True)

    # Combine price data and sentiment scores using outer join
    data = price_data.join(sentiment_data[['sentiment_compound']], how='outer')

    # Fill missing sentiment scores with default value (e.g., 0)
    data['sentiment_compound'].fillna(0, inplace=True)

    # Fill any remaining NaN values in price data
    data.fillna(method='ffill', inplace=True)
    data.fillna(method='bfill', inplace=True)

    return data

# Function to prepare data for LSTM/GRU
def prepare_lstm_data(data, seq_length, use_sentiment=True):
    if not use_sentiment:
        data = data.drop(columns=['sentiment_compound'])
```

```
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(data)

    X, y = [], []
    for i in range(seq_length, len(scaled_data)):
        X.append(scaled_data[i-seq_length:i])
        y.append(scaled_data[i, 0])  # Assuming 'Close' is the first column
    X, y = np.array(X), np.array(y)

    return X, y, scaler

# Function to plot results
def plot_results(train, test, predictions, model_name):
    plt.figure(figsize=(12, 6))
    plt.plot(train.index, train['Close'], label='Training Data')
    plt.plot(test.index, test['Close'], label='Actual Prices', color='blue')

    # Adjust predictions to align with test data indices
    predictions_index = test.index[:len(predictions)]
    plt.plot(predictions_index, predictions, label=f'{model_name} Predictions',
color='red')

    plt.legend()
    plt.show()

# Function to compute accuracy metrics
def compute_accuracy(y_true, y_pred):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mape = mean_absolute_percentage_error(y_true, y_pred)
    return rmse, mape

# Main function
def main():
    # Load data
    data = load_data('data/BTC-USD.csv',
'data/sentiment_analysis_output_advanced.csv')
    train_size = int(len(data) * 0.8)
    train, test = data.iloc[:train_size], data.iloc[train_size:]

    seq_length = 60

    # With sentiment data
    print("\nEvaluating models with sentiment data:")
    X, y, scaler = prepare_lstm_data(data, seq_length, use_sentiment=True)
    X_test, y_test = X[train_size-seq_length:], y[train_size-seq_length:]

    num_features = X_test.shape[2]
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], num_features))

    # Load trained LSTM model
    lstm_model = tf.keras.models.load_model('lstm_model.h5')

    # Load trained GRU model
    gru_model = tf.keras.models.load_model('gru_model.h5')

    # Predictions from LSTM and GRU models
    lstm_pred = lstm_model.predict(X_test)
    gru_pred = gru_model.predict(X_test)

    # Load the saved scaler parameters
    scaler = MinMaxScaler()
    scaler.scale_ = np.load('scaler.npy')
    scaler.min_ = np.load('scaler_min.npy')
    scaler.data_min_ = scaler.min_
    scaler.data_max_ = scaler.min_ + scaler.scale_

    # Inverse transform predictions to original scale
```

```
    lstm_pred = scaler.inverse_transform(np.hstack((lstm_pred,
np.zeros((lstm_pred.shape[0], num_features - 1)))))[:, 0]
    gru_pred = scaler.inverse_transform(np.hstack((gru_pred,
np.zeros((gru_pred.shape[0], num_features - 1)))))[:, 0]

    # Ensemble predictions using exponential formula
    alpha = 0.5
    ensemble_pred = alpha * lstm_pred + (1 - alpha) * gru_pred

    y_test_scaled = scaler.inverse_transform(np.hstack((y_test.reshape(-1, 1),
np.zeros((y_test.shape[0], num_features - 1)))))[:, 0]

    rmse, mape = compute_accuracy(y_test_scaled, ensemble_pred)
    print(f'With Sentiment - Ensemble RMSE: {rmse}')
    print(f'With Sentiment - Ensemble MAPE: {mape * 100}%')

    # Plot results for LSTM, GRU, and Ensemble with sentiment data
    plot_results(train, test, ensemble_pred, 'Ensemble with Sentiment')

if __name__ == "__main__":
    main()
```

Code for html pages

Index.html / HOMEPAGE:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>RapidEx - Buy & Sell Digital Assets In The RapidEx</title>

 <!-- favicon -->
 <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.svg') }}"
type="image/svg+xml">

 <!-- custom css link -->
 <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/style.css')
}}">
  <!--
    - google font link
  -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=DM+Sans:wght@400;500;700&display=swa
p" rel="stylesheet">
</head>

<body>

  <!--
    - #HEADER
  -->

  <header class="header" data-header>
    <div class="container">

      <a href="#" class="logo">
        <img src="{{ url_for('static', filename='assets/images/logo.svg') }}"
width="32" height="32" alt="RapidEx logo">
        RapidEx
      </a>
```

```
      <nav class="navbar" data-navbar>
        <ul class="navbar-list">

          <li class="navbar-item">
            <a href="/" class="navbar-link active" data-nav-link>Homepage</a>
          </li>


          <li class="navbar-item">
            <a href="/prediction" class="navbar-link" data-nav-link>Price
Prediction</a>
          </li>



        </ul>
      </nav>

      <button class="nav-toggle-btn" aria-label="Toggle menu" data-nav-toggler>
        <span class="line line-1"></span>
        <span class="line line-2"></span>
        <span class="line line-3"></span>
      </button>

      <a href="https://portfolio.metamask.io" class="btn btn-outline">Wallet</a>

    </div>
  </header>




  <main>
    <article>

      <!--
        - #HERO
      -->

      <section class="section hero" aria-label="hero" data-section>
        <div class="container">

          <div class="hero-content">

            <h1 class="h1 hero-title">Buy & Sell Digital Assets In The RapidEx</h1>

            <p class="hero-text">
              Coin RapidEx is the easiest, safest, and fastest way to buy & sell
crypto asset exchange.
            </p>

            <a href="#" class="btn btn-primary">Get started now</a>

          </div>

          <figure class="hero-banner">
            <img src=".{{ url_for('static', filename='assets/images/hero-
banner.png') }}" width="570" height="448" alt="hero banner" class="w-100">
          </figure>

        </div>
      </section>




      <!--
```

```
        - #TREND
      -->

      <section class="section trend" aria-label="crypto trend" data-section>
        <div class="container">

          <div class="trend-tab">

            <ul class="tab-nav">

              <li>
                <button class="tab-btn active">Crypto</button>
              </li>

              <li>
                <button class="tab-btn">DeFi</button>
              </li>

              <li>
                <button class="tab-btn">BSC</button>
              </li>

              <li>
                <button class="tab-btn">NFT</button>
              </li>

              <li>
                <button class="tab-btn">Metaverse</button>
              </li>

              <li>
                <button class="tab-btn">Polkadot</button>
              </li>

              <li>
                <button class="tab-btn">Solana</button>
              </li>

              <li>
                <button class="tab-btn">Opensea</button>
              </li>

              <li>
                <button class="tab-btn">Makersplace</button>
              </li>

            </ul>

            <ul class="tab-content">

              <li>
                <div class="trend-card">

                  <div class="card-title-wrapper">
                    <img src="{{ url_for('static', filename='assets/images/coin-
1.svg') }}" width="24" height="24" alt="bitcoin logo">

                    <a href="#" class="card-title">
                      Bitcoin <span class="span">BTC/USD</span>
                    </a>
                  </div>

                  <data class="card-value" value="46168.95">USD 46,168.95</data>

                  <div class="card-analytics">
                    <data class="current-price" value="36641.20">36,641.20</data>

                    <div class="badge red">-0.79%</div>
```

```
                </div>

              </div>
            </li>

            <li>
              <div class="trend-card active">

                <div class="card-title-wrapper">
                  <img src="{{ url_for('static', filename='assets/images/coin-
2.svg') }}" width="24" height="24" alt="ethereum logo">

                  <a href="#" class="card-title">
                    Ethereum <span class="span">ETH/USD</span>
                  </a>
                </div>

                <data class="card-value" value="3480.04">USD 3,480.04</data>

                <div class="card-analytics">
                  <data class="current-price" value="36641.20">36,641.20</data>

                  <div class="badge green">+10.55%</div>
                </div>

              </div>
            </li>

            <li>
              <div class="trend-card">

                <div class="card-title-wrapper">
                  <img src="{{ url_for('static', filename='assets/images/coin-
3.svg') }}" width="24" height="24" alt="tether logo">

                  <a href="#" class="card-title">
                    Tether <span class="span">USDT/USD</span>
                  </a>
                </div>

                <data class="card-value" value="1.00">USD 1.00</data>

                <div class="card-analytics">
                  <data class="current-price" value="36641.20">36,641.20</data>

                  <div class="badge red">-0.01%</div>
                </div>

              </div>
            </li>

            <li>
              <div class="trend-card">

                <div class="card-title-wrapper">
                  <img src="{{ url_for('static', filename='assets/images/coin-
4.svg') }}" width="24" height="24" alt="bnb logo">

                  <a href="#" class="card-title">
                    BNB <span class="span">BNB/USD</span>
                  </a>
                </div>

                <data class="card-value" value="443.56">USD 443.56</data>

                <div class="card-analytics">
                  <data class="current-price" value="36641.20">36,641.20</data>
```

```
                <div class="badge red">-1.24%</div>
              </div>

            </div>
          </li>

        </ul>

      </div>

    </div>
  </section>




  <!--
    - #MARKET
  -->

  <section class="section market" aria-label="market update" data-section>
    <div class="container">

      <div class="title-wrapper">
        <h2 class="h2 section-title">Market Update</h2>

        <a href="#" class="btn-link">See All Coins</a>
      </div>

      <div class="market-tab">

        <ul class="tab-nav">

          <li>
            <button class="tab-btn active">View All</button>
          </li>

          <li>
            <button class="tab-btn">Metaverse</button>
          </li>

          <li>
            <button class="tab-btn">Entertainment</button>
          </li>

          <li>
            <button class="tab-btn">Energy</button>
          </li>

          <li>
            <button class="tab-btn">NFT</button>
          </li>

          <li>
            <button class="tab-btn">Gaming</button>
          </li>

          <li>
            <button class="tab-btn">Music</button>
          </li>

        </ul>

        <table class="market-table">

          <thead class="table-head">

            <tr class="table-row table-title">
```

```
                  <th class="table-heading"></th>

                  <th class="table-heading" scope="col">#</th>

                  <th class="table-heading" scope="col">Name</th>

                  <th class="table-heading" scope="col">Last Price</th>

                  <th class="table-heading" scope="col">24h %</th>

                  <th class="table-heading" scope="col">Market Cap</th>

                  <th class="table-heading" scope="col">Last 7 Days</th>

                  <th class="table-heading"></th>

                </tr>

              </thead>

              <tbody class="table-body">

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">1</th>

                  <td class="table-data">
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
1.svg') }}" width="20" height="20" alt="Bitcoin logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">Bitcoin <span
class="span">BTC</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>

                  <td class="table-data last-update green">+1.45%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
1.svg') }}" width="100" height="40" alt="profit chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">
```

```
                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">2</th>

                  <td class="table-data">
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
1.svg') }}" width="20" height="20" alt="Ethereum logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">Ethereum <span
class="span">ETH</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>

                  <td class="table-data last-update red">-5.12%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
2.svg') }}" width="100" height="40" alt="loss chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">3</th>

                  <td class="table-data">
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
3.svg') }}" width="20" height="20" alt="Tether logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">Tether <span
class="span">USDT/USD</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>
```

```
                  <td class="table-data last-update green">+1.45%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
1.svg') }}" width="100" height="40" alt="profit chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">4</th>

                  <td class="table-data">
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
4.svg') }}" width="20" height="20" alt="BNB logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">BNB <span
class="span">BNB/USD</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>

                  <td class="table-data last-update red">-3.75%%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
2.svg') }}" width="100" height="40" alt="loss chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
```

```
                  </button>
                </td>

                <th class="table-data rank" scope="row">5</th>

                <td class="table-data">
                  <div class="wrapper">
                    <img src="{{ url_for('static', filename='assets/images/coin-
5.svg') }}" width="20" height="20" alt="Solana logo" class="img">

                    <h3>
                      <a href="#" class="coin-name">Solana <span
class="span">SOL</span></a>
                    </h3>
                  </div>
                </td>

                <td class="table-data last-price">$56,623.54</td>

                <td class="table-data last-update green">+1.45%</td>

                <td class="table-data market-cap">$880,423,640,582</td>

                <td class="table-data">
                  <img src="{{ url_for('static', filename='assets/images/chart-
1.svg') }}" width="100" height="40" alt="profit chart" class="chart">
                </td>

                <td class="table-data">
                  <button class="btn btn-outline">Trade</button>
                </td>

              </tr>

              <tr class="table-row">

                <td class="table-data">
                  <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                    <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                    <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                  </button>
                </td>

                <th class="table-data rank" scope="row">6</th>

                <td class="table-data">
                  <div class="wrapper">
                    <img src="{{ url_for('static', filename='assets/images/coin-
6.svg') }}" width="20" height="20" alt="XRP logo" class="img">

                    <h3>
                      <a href="#" class="coin-name">XRP <span
class="span">XRP</span></a>
                    </h3>
                  </div>
                </td>

                <td class="table-data last-price">$56,623.54</td>

                <td class="table-data last-update red">-2.22%</td>

                <td class="table-data market-cap">$880,423,640,582</td>

                <td class="table-data">
```

```
                    <img src="{{ url_for('static', filename='assets/images/chart-
2.svg') }}" width="100" height="40" alt="loss chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">7</th>

                  <td class="table-data">
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
7.svg') }}" width="20" height="20" alt="Cardano logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">Cardano <span
class="span">ADA</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>

                  <td class="table-data last-update green">+0.8%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
2.svg') }}" width="100" height="40" alt="profit chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

                <tr class="table-row">

                  <td class="table-data">
                    <button class="add-to-fav" aria-label="Add to favourite" data-
add-to-fav>
                      <ion-icon name="star-outline" aria-hidden="true" class="icon-
outline"></ion-icon>
                      <ion-icon name="star" aria-hidden="true" class="icon-
fill"></ion-icon>
                    </button>
                  </td>

                  <th class="table-data rank" scope="row">8</th>

                  <td class="table-data">
```

```
                    <div class="wrapper">
                      <img src="{{ url_for('static', filename='assets/images/coin-
8.svg') }}" width="20" height="20" alt="Avalanche logo" class="img">

                      <h3>
                        <a href="#" class="coin-name">Avalanche <span
class="span">AVAX</span></a>
                      </h3>
                    </div>
                  </td>

                  <td class="table-data last-price">$56,623.54</td>

                  <td class="table-data last-update green">+1.41%</td>

                  <td class="table-data market-cap">$880,423,640,582</td>

                  <td class="table-data">
                    <img src="{{ url_for('static', filename='assets/images/chart-
1.svg') }}" width="100" height="40" alt="profit chart" class="chart">
                  </td>

                  <td class="table-data">
                    <button class="btn btn-outline">Trade</button>
                  </td>

                </tr>

              </tbody>

            </table>

          </div>

        </div>
      </section>




      <!--
        - #INSTRUCTION
      -->




      <!--
        - #ABOUT
      -->

      <section class="section about" aria-label="about" data-section>
        <div class="container">

          <figure class="about-banner">
            <img src="{{ url_for('static', filename='assets/images/about-
banner.png') }}" width="748" height="436" loading="lazy" alt="about banner"
              class="w-100">
          </figure>

          <div class="about-content">

            <h2 class="h2 section-title">What Is RapidEx</h2>

            <p class="section-text">
              Experience a variety of trading on Bitcost. You can use various types
of coin transactions such as Spot
```

```
              Trade, Futures
              Trade, P2P, Staking, Mining, and margin.
            </p>

            <ul class="section-list">

              <li class="section-item">
                <div class="title-wrapper">
                  <ion-icon name="checkmark-circle" aria-hidden="true"></ion-icon>

                  <h3 class="h3 list-title">View real-time cryptocurrency
prices</h3>
                </div>

                <p class="item-text">
                  Experience a variety of trading on Bitcost. You can use various
types of coin transactions such as
                  Spot Trade, Futures
                  Trade, P2P, Staking, Mining, and margin.
                </p>
              </li>

              <li class="section-item">
                <div class="title-wrapper">
                  <ion-icon name="checkmark-circle" aria-hidden="true"></ion-icon>

                  <h3 class="h3 list-title">Buy and sell BTC, ETH, XRP, OKB,
Etc...</h3>
                </div>

                <p class="item-text">
                  Experience a variety of trading on Bitcost. You can use various
types of coin transactions such as
                  Spot Trade, Futures
                  Trade, P2P, Staking, Mining, and margin.
                </p>
              </li>

            </ul>

            <a href="#" class="btn btn-primary">Explore More</a>

          </div>

        </div>
      </section>




      <!--
        - #APP
      -->



    </article>
  </main>




  <!--
    - #FOOTER
  -->

  <footer class="footer">
```

```
    <div class="footer-top" data-section>
      <div class="container">

        <div class="footer-brand">

          <a href="#" class="logo">
            <img src="{{ url_for('static', filename='assets/images/logo.svg') }}"
width="50" height="50" alt="RapidEx logo">
            RapidEx
          </a>

          <h2 class="footer-title">Let's talk! 💪</h2>

          <a href="tel:+123456789101" class="footer-contact-link">+91
6360732957</a>

          <a href="mailto:hello.RapidEx@gmail.com" class="footer-contact-
link">sujay0620@gmail.com</a>

          <address class="footer-contact-link">
            Bengaluru, Karnataka
          </address>

        </div>

        <ul class="footer-list">

          <li>
            <p class="footer-list-title">Products</p>
          </li>

          <li>
            <a href="#" class="footer-link">Spot</a>
          </li>

          <li>
            <a href="#" class="footer-link">Inverse Perpetual</a>
          </li>

          <li>
            <a href="#" class="footer-link">USDT Perpetual</a>
          </li>

          <li>
            <a href="#" class="footer-link">Exchange</a>
          </li>

          <li>
            <a href="#" class="footer-link">Launchpad</a>
          </li>

          <li>
            <a href="#" class="footer-link">Binance Pay</a>
          </li>

        </ul>

        <ul class="footer-list">

          <li>
            <p class="footer-list-title">Services</p>
          </li>

          <li>
            <a href="#" class="footer-link">Buy Crypto</a>
          </li>

          <li>
```

```
            <a href="#" class="footer-link">Markets</a>
          </li>

          <li>
            <a href="#" class="footer-link">Tranding Fee</a>
          </li>

          <li>
            <a href="#" class="footer-link">Affiliate Program</a>
          </li>

          <li>
            <a href="#" class="footer-link">Referral Program</a>
          </li>

          <li>
            <a href="#" class="footer-link">API</a>
          </li>

        </ul>

        <ul class="footer-list">

          <li>
            <p class="footer-list-title">Support</p>
          </li>

          <li>
            <a href="#" class="footer-link">Bybit Learn</a>
          </li>

          <li>
            <a href="#" class="footer-link">Help Center</a>
          </li>

          <li>
            <a href="#" class="footer-link">User Feedback</a>
          </li>

          <li>
            <a href="#" class="footer-link">Submit a request</a>
          </li>

          <li>
            <a href="#" class="footer-link">API Documentation</a>
          </li>

          <li>
            <a href="#" class="footer-link">Trading Rules</a>
          </li>

        </ul>

        <ul class="footer-list">

          <li>
            <p class="footer-list-title">About Us</p>
          </li>

          <li>
            <a href="#" class="footer-link">About Bybit</a>
          </li>

          <li>
            <a href="#" class="footer-link">Authenticity Check</a>
          </li>

          <li>
```

```
            <a href="#" class="footer-link">Careers</a>
          </li>

          <li>
            <a href="#" class="footer-link">Business Contacts</a>
          </li>

          <li>
            <a href="#" class="footer-link">Blog</a>
          </li>

        </ul>

      </div>
    </div>

    <div class="footer-bottom">
      <div class="container">

        <p class="copyright">
          &copy; 2022 Rapid All Rights Reserved by <a href="#" class="copyright-
link">sujay0620</a>
        </p>

        <ul class="social-list">

          <li>
            <a href="#" class="social-link">
              <ion-icon name="logo-facebook"></ion-icon>
            </a>
          </li>

          <li>
            <a href="#" class="social-link">
              <ion-icon name="logo-twitter"></ion-icon>
            </a>
          </li>

          <li>
            <a href="#" class="social-link">
              <ion-icon name="logo-instagram"></ion-icon>
            </a>
          </li>

          <li>
            <a href="#" class="social-link">
              <ion-icon name="logo-linkedin"></ion-icon>
            </a>
          </li>

        </ul>

      </div>
    </div>

  </footer>




  <!--
    - custom js link
  -->
  <script src="{{ url_for('static', filename='assets/js/script.js') }}"
defer></script>

  <!--
    - ionicon link
```

```
  -->
  <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
  <script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
  <script>
    async function getPrediction() {
      const prices =
document.getElementById('prices').value.split(',').map(Number);
      const response = await fetch('/predict', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ prices }),
      });
      const data = await response.json();
      document.getElementById('prediction').innerText = `Predicted Price:
${data.prediction}`;
    }
  </script>
</body>

</html>
```

Code for Price prediction HTML page

prediction.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>RapidEx - Buy & Sell Digital Assets In The RapidEx</title>

  <!-- favicon -->
  <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.svg') }}"
type="image/svg+xml">

  <!-- custom css link -->
  <link rel="stylesheet" href="{{ url_for('static',
filename='assets/css/style.css') }}">

  <!-- google font link -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=DM+Sans:wght@400;500;700&display=swa
p" rel="stylesheet">
  <style>
    #results {
      opacity: 0;
      transition: opacity 1s ease-in-out;
    }
    #results.visible {
      opacity: 1;
    }
  </style>
</head>

<body>
  <header class="header" data-header>
    <div class="container">
```

```
      <a href="#" class="logo">
        <img src="{{ url_for('static', filename='assets/images/logo.svg') }}"
width="32" height="32" alt="RapidEx logo">
        RapidEx
      </a>

      <nav class="navbar" data-navbar>
        <ul class="navbar-list">
          <li class="navbar-item">
            <a href="/" class="navbar-link" data-nav-link>Homepage</a>
          </li>
          <li class="navbar-item">
            <a href="/prediction" class="navbar-link active" data-nav-link>Price
Prediction</a>
          </li>
        </ul>
      </nav>

      <button class="nav-toggle-btn" aria-label="Toggle menu" data-nav-toggler>
        <span class="line line-1"></span>
        <span class="line line-2"></span>
        <span class="line line-3"></span>
      </button>

      <a href="https://portfolio.metamask.io" class="btn btn-outline">Wallet</a>
    </div>
  </header>

  <main>
    <div class="container">
      <h2 class="h2 section-title">Cryptocurrency Price Prediction</h2>
      <form id="predictForm" enctype="multipart/form-data">
        <label for="crypto">Select Cryptocurrency:</label>
        <select id="crypto" name="crypto">
          <option value="BTC">Bitcoin (BTC)</option>
          <option value="ETH">Ethereum (ETH)</option>
          <option value="LTC">Litecoin (LTC)</option>
        </select>

        <label for="model">Select Model:</label>
        <select id="model" name="model">
          <option value="ANN">ANN</option>
          <option value="LSTM">LSTM</option>
          <option value="LSTM-GRU">LSTM-GRU</option>
          <option value="GBM">GBM</option>
          <option value="RFR">Random Forest</option>
        </select>

        <label for="seq_length">Sequence Length For Prediction:</label>
        <input type="number" id="seq_length" name="seq_length" min="1" value="60">

        <label for="csvFile">Upload a CSV file ({crypto}-USD.csv from yahoo
finance):</label>
        <input type="file" id="csvFile" name="csvFile" accept=".csv">
        <button type="button" onclick="uploadCSV()">Upload & Predict</button>
      </form>
      <div id="results"></div>
      <div id="loading" style="display: none;">
        <img src="{{ url_for('static', filename='assets/images/loading.gif') }}"
alt="Loading..." width="60" height="60" style="padding-left: 20px;">
      </div>
    </div>
  </main>

  <script>
    function uploadCSV() {
      const formData = new FormData(document.getElementById('predictForm'));
      document.getElementById('loading').style.display = 'block';
```

```
      document.getElementById('results').innerHTML = '';
      document.getElementById('results').classList.remove('visible');

      fetch('/predict', {
        method: 'POST',
        body: formData
      })
      .then(response => response.json())
      .then(data => {
        document.getElementById('loading').style.display = 'none';
        document.getElementById('results').innerHTML = `
          <p>Root Mean Square Error (RMSE): ${data.mse}</p>
          <p>Mean Absolute Percentage Error (MAPE): ${data.mape * 100}%</p>
          <p>Real v/s Prediction Plot</p>
          <img src="${data.plot_url}" alt="Prediction Plot">
        `;
        document.getElementById('results').classList.add('visible');
      })
      .catch(error => {
        document.getElementById('loading').style.display = 'none';
        console.error('Error:', error);
      });
    }
  </script>

  <script src="{{ url_for('static', filename='assets/js/script.js') }}"
defer></script>

  <!-- ionicon link -->
  <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
  <script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>

</body>

</html>
```