Big Data Analytics

A Project Submitted

By

Batch 2

Under the Guidance of

Kunal

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT

YELAHANKA BANGALORE

Phase 3 Documentation

Project: Big Data Analytics

Phase 3 Submission by: Batch 2

Year: 4th Year, Computer Science and Engineering (CSE)

1 Introduction

The development phase is crucial in transforming raw data into actionable insights using machine learning models. In this project, we aimed to build a rainfall prediction system based on environmental parameters such as temperature, humidity, pressure, and wind speed. The objective was to classify whether rainfall would occur (Yes or No) using three distinct models: Random Forest, Logistic Regression, and LSTM Neural Networks.

2 Data Collection and Loading

- The dataset was collected and stored in a CSV file named rainfall.csv.
- Pandas was used to load the data into a DataFrame:

```
df = pd.read_csv('rainfall.csv')
```

- Initial exploration included displaying the head of the dataset and checking for null values using:

```
df.head()
df.isnull().sum()
```

3 Data Cleaning and Preprocessing

- **Dropping Missing Values**: All rows containing null values were removed using:

```
df = df.dropna()
```

- **Dropping Irrelevant Columns**: Certain columns like Unnamed: 0, SUBDIVISION, and YEAR were considered irrelevant and

dropped:

```
df = df.drop(['Unnamed: 0', 'SUBDIVISION', 'YEAR'], axis=1)
```

- **Encoding Categorical Labels**: The target variable "Rainfall" was encoded using LabelEncoder() to convert Yes/No into 1/0:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['Rainfall'] = encoder.fit_transform(df['Rainfall'])
```

4 Feature Selection and Splitting

- **Input Features (X)**:
  - 'TEMPERATURE'
  - 'HUMIDITY'
  - 'PRESSURE'
  - 'WINDSPEED'
- **Target Variable (y)**:
  - 'Rainfall' (binary encoded)

```
X = df[['TEMPERATURE', 'HUMIDITY', 'PRESSURE', 'WINDSPEED']]
y = df['Rainfall']
```

- **Train-Test Split**:
  - 80% for training, 20% for testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

5 Model 1: Random Forest Classifier

- **Model**: RandomForestClassifier() from Scikit-learn.
- **Training**:

```
model1 = RandomForestClassifier()
model1.fit(X_train, y_train)
```

- **Evaluation**:

```
y_pred = model1.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

- **Outcome**: Provided a strong baseline performance due to its ensemble nature and resistance to overfitting.

6 Model 2: Logistic Regression

- **Model**: LogisticRegression() from Scikit-learn.
- **Training**:

```
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

Evaluation:

```
y_pred2 = model2.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred2))
```

- **Outcome**: Offered a simple, interpretable model for binary classification.

7 Model 3: LSTM Neural Network

- **Model Type**: Long Short-Term Memory (LSTM) using TensorFlow/Keras.
- **Preprocessing**:
  - **Scaling**: Input features were normalized using MinMaxScaler():

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

  - **Reshaping for LSTM**:

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

- **Model Building**:

```
model3 = Sequential()
model3.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2])))
model3.add(Dense(1, activation='sigmoid'))
model3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- **Training**:

```
model3.fit(X_train, y_train, epochs=10, verbose=1)
```

- **Evaluation**:

```
_, accuracy = model3.evaluate(X_test, y_test, verbose=0)
print("LSTM Accuracy:", accuracy)
```

- **Outcome**: Showcased how deep learning models can learn from environmental sequences and patterns.

8 Summary of Development

- The code demonstrated complete steps from data ingestion to model training and evaluation.
- Multiple models were developed to assess performance trade-offs.
- Preprocessing, scaling, and reshaping were carefully handled, especially for deep learning.
- This setup allows the system to be modular and ready for deployment, integration with APIs, or front-end interfaces.