DOCUMENTATION FOR RAINFALL PREDICTION MODEL

Introduction

Today there are no certain methods by using which we can predict whether there will be rainfall today or not. Even the meteorological department's prediction fails sometimes. In this article, we will learn how to build a machine-learning model which can predict whether there will be rainfall today or not based on some atmospheric factors. This problem is related to Rainfall Prediction using Machine Learning because machine leaning models tend to perform better on the previously known task which needed highly skilled individuals to do so. In this project, we'll create a machine learning model which will be trained on the datasets so that it can predict the rainfall. Various steps which involves while creating this model includes data pre-processing, handled imbalanced data, train model and then evaluate its performance.

Objective of the Project

The primary objective of this project is to develop a machine learning model capable of accurately predicting whether it will rain tomorrow based on historical weather data. This involves data pre-processing, handling imbalanced datasets, feature engineering, and training multiple machine learning models. The goal is to evaluate and select the best-performing model to ensure reliable rainfall predictions, which can be used for better planning and decision-making.

Datasets

The dataset is a modified version of a rainfall prediction dataset taken from kaggle and updated with weather's now website which contains various weather-related features along with a target variable indicating whether it rained or not. Day represents the specific day the weather data corresponds to (this column was later removed as it was deemed unnecessary for prediction), rainfall is the target variable indicating whether it rained on that day ('yes' or 'no'). It also includes various factors such as temperature (max and min), humidity, pressure, wind speed, and possibly others relevant to weather prediction. Specifically, maxtemp and mintemp were mentioned, though they were removed due to high correlation with other features.

Steps involves in model creation

1.Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay

import warnings
warnings.filterwarnings('ignore')
```

We are importing various libraries for data manipulation, visualization, model building, and evaluation. Python libraries make it easy for us to handle the data and perform typical and complex tasks with a single line of code.

- **Pandas** – This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.
- **Numpy** – Numpy arrays are very fast and can perform large computations in a very short time.
- **Matplotlib/Seaborn** – This library is used to draw visualizations.
- **Sklearn** – This module contains multiple libraries are having pre-implemented functions to perform tasks from data pre-processing to model development and evaluation.
- **XGBoost** – This contains the extreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.
- **Imblearn** – This module contains a function that can be used for handling problems related to data imbalance.

2. Loading and Exploring the Data:

```
df = pd.read_csv("C:\\Users\Rishabh Kumar\\Downloa
df.head()
df.shape
df.info
df.describe()
```

- we load the dataset modified_rainfall_prediction_data.csv.
- we perform initial data exploration by displaying the first few rows, checking the shape, data types, summary statistics, and null values of the dataset.

3. Null Value Imputation:

```
#Null value imputation
for col in df.columns:

# Checking if the column contains any null values
    if df[col].isnull().sum() > 0:
        val = df[col].mean();df[col] = df[col].fillna(val)

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

- we handle missing values by filling them with the mean of the respective columns.

4. Data Preprocessing:

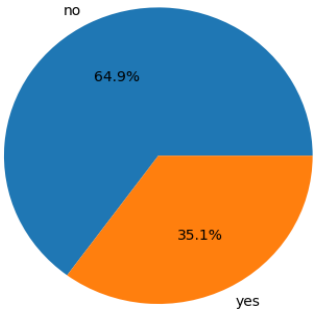
```
#Removing unnecessary day column from the datasets
features = list(df.select_dtypes(include=np.number).columns)
features.remove('day')
print(features)
```

- We remove the unnecessary day column from the dataset.

```
#Replacing yes with 1 and no with 0 in rainfall columns.
df.replace({'yes':1, 'no':0}, inplace=True)
```

- We encode the target variable (rainfall) by replacing 'yes' with 1 and 'no' with 0.

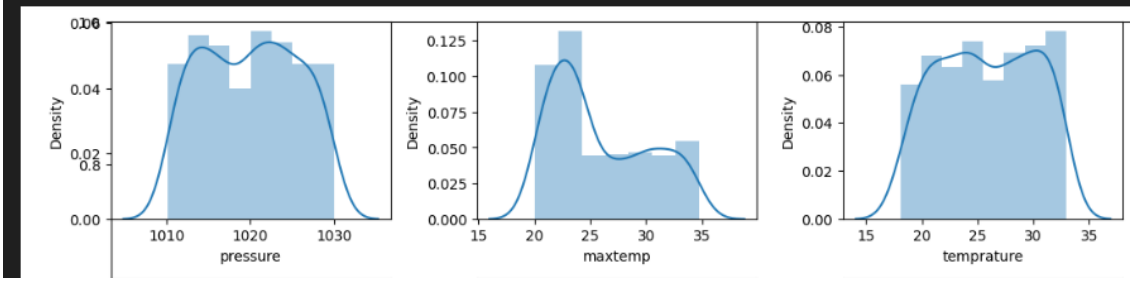
5. Exploratory Data Analysis (EDA):



- we visualize the distribution of the target variable (rainfall) using a pie chart.
- we compute the mean of the features grouped by the target variable.
- we visualize the distribution and box plots of the numerical features to understand their distributions and detect outliers.

```
plt.subplots(figsize = (15,8))
for i, col in enumerate(features):
    plt.subplot(3,4,i+1)
    sb.distplot(df[col])
plt.tight_layout()
plt.show()
```

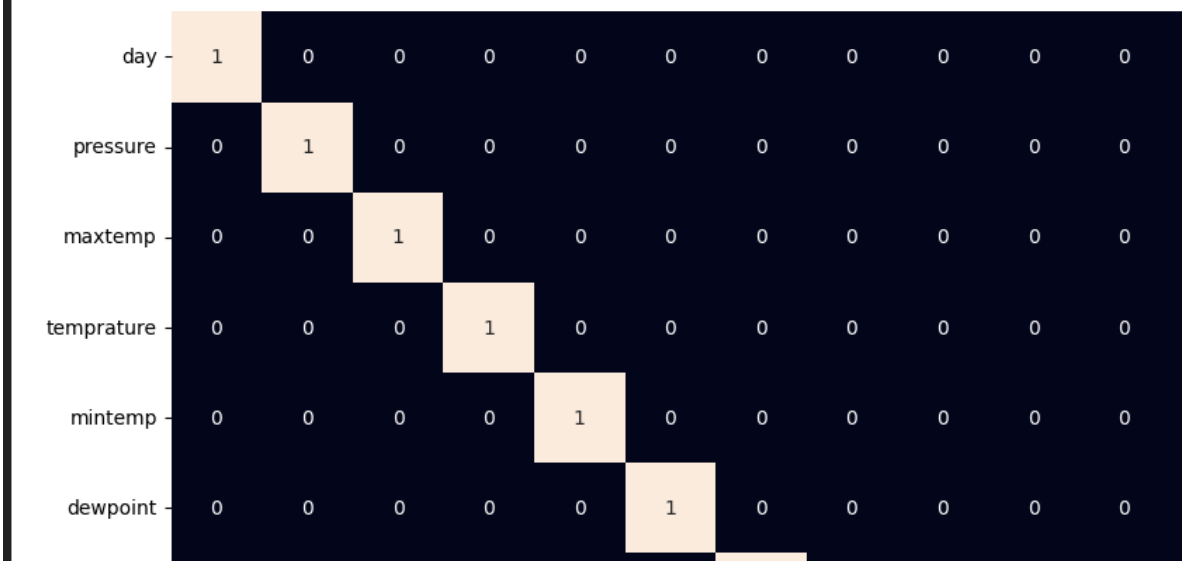
✓ 2.0s



- we check for correlations between the features using a heatmap and decide to drop highly correlated features (maxtemp and mintemp).

```
#Checking for correlation using heatmap
plt.figure(figsize=(10,10))
sb.heatmap(df.corr() > 0.8,annot=True,cbar=False)
plt.show()
```

✓ 0.5s



6. Splitting the Data:

```
X_train, X_val,Y_train, Y_val = train_test_split(features,target,test_size=0.2,stratify=target,random_state=2)

# As the data was highly imbalanced we will
# balance it by adding repetitive rows of minority class.
ros = RandomOverSampler(sampling_strategy='minority',random_state=22)
X, Y = ros.fit_resample(X_train, Y_train)
```

- We split the dataset into training and validation sets using train_test_split, ensuring that the split maintains the class distribution (stratify=target).

7. Model Training:

```
models = [LogisticRegression(), XGBClassifier(), SVC(kernel='rbf', probability=True)]

for model in models:
    model.fit(X, Y)

    model_name = type(model).__name__
    print(f'{model_name} :')

    train_preds = model.predict_proba(X)
    train_accuracy = metrics.roc_auc_score(Y, train_preds[:, 1])
    print(f'Training Accuracy :  {train_accuracy}')

    val_preds = model.predict_proba(X_val)
    val_accuracy = metrics.roc_auc_score(Y_val, val_preds[:, 1])
    print(f'Validation Accuracy :  {val_accuracy}')
    print()
```

- We train three different models: Logistic Regression (LogisticRegression), XGBoost Classifier (XGBClassifier), and Support Vector Classifier (SVC).
- For each model, we fit it to the resampled training data.

8. Model Evaluation:

```
LogisticRegression :
Training Accuracy :  0.9822714681440443
Validation Accuracy :  0.9885433715220949

XGBClassifier :
Training Accuracy :  1.0
Validation Accuracy :  0.9877250409165304

SVC :
Training Accuracy :  0.995096952908587 2
Validation Accuracy :  0.997545008183306
```

- We evaluate the models using the ROC AUC score on both the training and validation sets to assess their performance.

Summary

Overall, this project aims to develop a machine learning model to predict whether it will rain tomorrow using historical weather data. The dataset includes various weather features and the target variable, rainfall. Key steps involve data pre-processing, including handling missing values and balancing the imbalanced target variable using oversampling. Exploratory data analysis helps in understanding feature distributions and correlations. After normalizing the features, multiple models (Logistic Regression, XGBoost, SVM) are trained and evaluated using ROC AUC scores. The goal is to select the best-performing model for accurate rainfall prediction, aiding in better planning and decision-making.