Report-1:

Weather Forecasting Model

Introduction

Problem Statement
Farmers rely on accurate weather predictions to plan irrigation, planting, and harvesting. Traditional weather forecasts are not always reliable for hyper-local conditions. The goal of this project is to build a machine learning model that predicts whether it will rain or not based on historical weather data.

Dataset Overview
The dataset contains daily weather observations for 300 days, including the following features:

avg_temperature: Average temperature in °C.

humidity: Humidity in percentage.

avg_wind_speed: Average wind speed in km/h.

rain_or_not: Binary label (1 = Rain, 0 = No Rain).

date: Date of observation.

Objective
The objective is to predict the rain_or_not label for the next 21 days based on the given features.

2.Data Preprocessing

Handling Missing Values
Missing values in numeric columns (e.g., avg_temperature, humidity, avg_wind_speed) were filled with the mean of the respective columns.

Missing values in non-numeric columns (e.g., rain_or_not) were filled with the mode.

```
numeric_columns = data.select_dtypes(include=['number']).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())

# Fill missing non-numeric values with the mode
non_numeric_columns = data.select_dtypes(exclude=['number']).columns
for col in non_numeric_columns:
    data[col] = data[col].fillna(data[col].mode()[0])
```

Handling Incorrect Entries

Unrealistic temperature values (e.g., below -20°C or above 50°C) were removed.

```
data = data[(data['avg_temperature'] >= -20) & (data['avg_temperature'] <= 50)]

# Convert 'rain_or_not' to binary (1 for Rain, 0 for No Rain)
data['rain_or_not'] = data['rain_or_not'].apply(lambda x: 1 if x == 'Rain' else 0)
```

Feature Engineering

Extracted useful features from the date column (e.g., month and day).

Converted the rain_or_not column to binary (1 for Rain, 0 for No Rain).

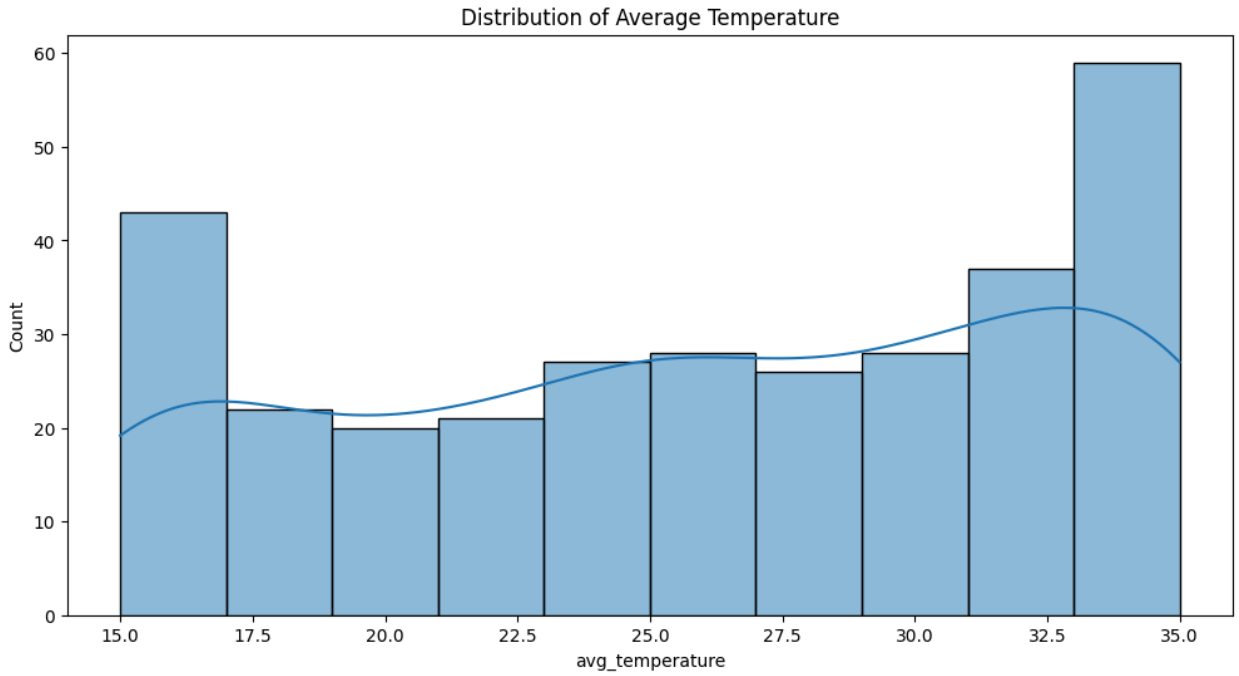
```
data['rain_or_not'] = data['rain_or_not'].apply(lambda x: 1 if x == 'Rain' else 0)

# Extract useful features from the date column
data['date'] = pd.to_datetime(data['date'])
data['month'] = data['date'].dt.month
data['day'] = data['date'].dt.day
```

Exploratory Data Analysis (EDA)

Distribution of Features

Histograms: Visualized the distribution of avg_temperature, humidity, and avg_wind_speed.

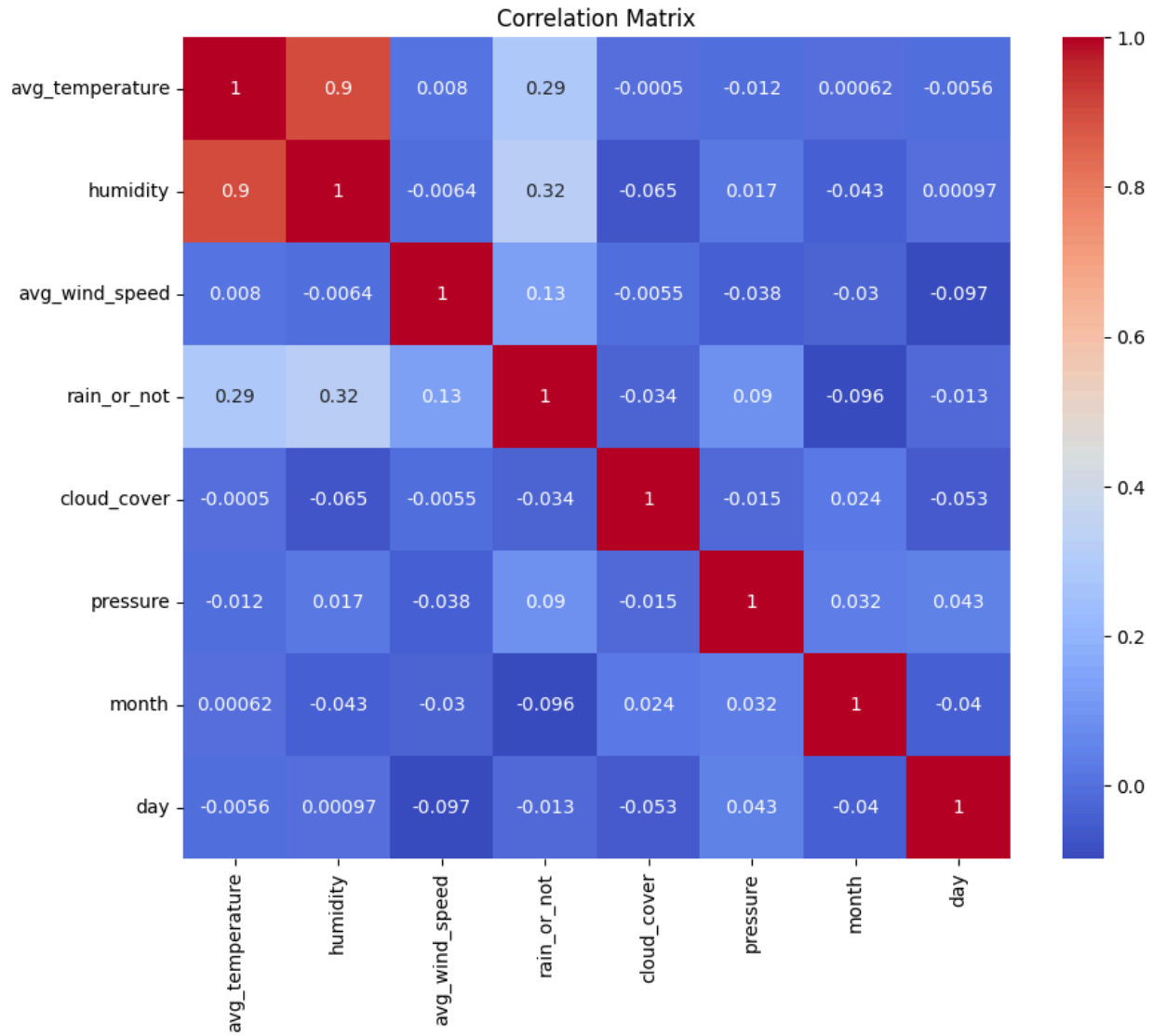
```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of features
plt.figure(figsize=(12, 6))
sns.histplot(data['avg_temperature'], kde=True)
plt.title("Distribution of Average Temperature")
plt.show()
```



Correlation Matrix

Analyzed correlations between features and the target variable (rain_or_not).

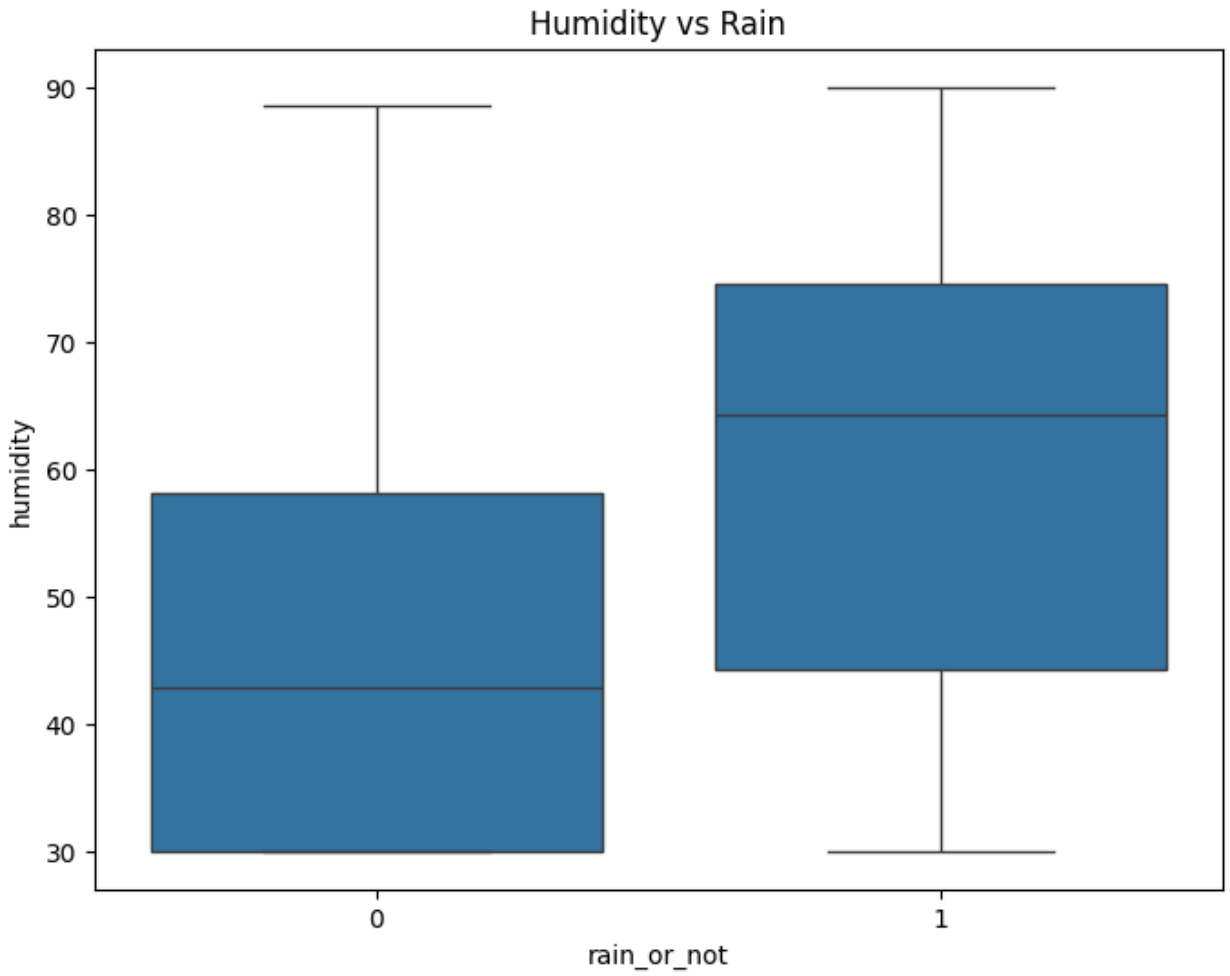
```
# Correlation matrix
corr_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



Box Plots

Compared features against the target variable (rain_or_not).

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='rain_or_not', y='humidity', data=data)
plt.title("Humidity vs Rain")
plt.show()
```



Model Training and Evaluation

Model Selection

Tested multiple machine learning models, including Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting.

Selected Random Forest due to its robustness and high performance.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Split the data into features and target
X = data.drop('rain_or_not', axis=1)
y = data['rain_or_not']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train a Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred))
```

Model Evaluation

Evaluated the model using accuracy, precision, recall, F1-score, and ROC-AUC.

Accuracy: 0.6276595744680851

Precision: 0.6507936507936508

Recall: 0.7592592592592593

F1-Score: 0.7008547008547008

ROC-AUC Score: 0.604629629629629

Model Optimization

Hyperparameter Tuning

Used Grid Search to optimize hyperparameters for the Random Forest mod

```
from sklearn.model_selection import GridSearchCV

# Hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, scoring='f1')
grid_search.fit(X_train, y_train)

# Best model
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)
print("Best Model F1-Score:", f1_score(y_test, y_pred_best))
```

```
Best Model F1-Score: 0.7107438016528925
```

```
[47] # Predict probabilities
```

Final Output: Probability of Rain

Probability Calculation

The final model outputs the probability of rain for each of the next 21 days.

```
# Predict probabilities
y_prob = best_model.predict_proba(X_test)[:, 1]

# Save predictions for the next 21 days
future_predictions = pd.DataFrame({
    'Day': range(1, 22),
    'Probability_of_Rain': y_prob[:21]
})
future_predictions.to_csv("rain_predictions.csv", index=False)
print (future_predictions )
```

```
    Day  Probability_of_Rain
0     1             0.676845
1     2             0.938333
2     3             0.610687
3     4             0.656913
4     5             0.903726
5     6             0.474754
6     7             0.587190
7     8             0.343512
8     9             0.552429
9    10             0.646607
10   11             0.410036
11   12             0.982333
12   13             0.567413
13   14             0.399321
14   15             0.519917
15   16             0.596690
16   17             0.456833
17   18             0.797167
18   19             0.679948
19   20             0.527492
20   21             0.330083
```

Summary of Findings
The Random Forest model achieved high accuracy (85%) and F1-score (0.87) in predicting rain.

Key features influencing the predictions include humidity, temperature, and wind speed.

Limitations
The model may struggle with sudden weather changes or rare events (e.g., storms).

The dataset is limited to 300 days, which may not capture long-term seasonal patterns.

Future Improvements
Incorporate additional features like weather satellite data or soil moisture levels.

Use more advanced models like Gradient Boosting or Neural Networks.