WEATHER PREDICTION SYSTEM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

NANDHAKUMAR A [RA2111047010007]
VIDHYUTH KASTHURIRANGAN [RA2111047010016]

Under the guidance of

Dr. Karpagam

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"WEATHER PREDICTION SYSTEM"** is the bona fide work of **NANDHAKUMAR A [RA2111047010007], VIDHYUTH KASTHURIRANGAN [RA2111047010016]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Karpagam
Assistant Professor
Department of Computer Science and Engineering

ABSTRACT

The average temperature forecasted by the weather prediction model in this report is intended to be applied to various areas in Jordan. The model starts with preprocessing the data to deal with irregularities and missing values. Then, it uses feature engineering to extract important variables that affect temperature fluctuations. Then, different machine learning methods are used to predict temperatures, such as Decision Tree, Random Forest, Polynomial Regression, Linear Regression, and Bagging Regressor. To evaluate each model's accuracy and effectiveness, performance evaluation metrics including R-squared (R2) scores and root mean square error (RMSE) are used. The performance of the weather prediction model and its possible uses in meteorological forecasting are thoroughly examined in this research.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CSV	Comma Separated Values
RMSE	Root Mean Squared Error
EDA	Exploratory Data Analysis

CHAPTER 1
INTRODUCTION

Weather forecasting is a crucial task that has significant implications for various sectors, including agriculture, transportation, energy management, and public safety. Accurate weather predictions can help mitigate the adverse effects of extreme weather conditions and facilitate better decision-making processes. With the advent of machine learning techniques, weather forecasting has gained new momentum, as these techniques can effectively analyze and learn from complex weather data patterns.

The provided code is a comprehensive data analysis and machine learning pipeline designed to predict average temperatures using historical weather data. The dataset used in this analysis is obtained from a CSV file named "weather_data_24hr.csv," which contains various weather observations, including temperature, precipitation, wind speed, humidity, and other meteorological measurements.

The code follows a structured approach, beginning with data preprocessing, exploratory data analysis, and feature engineering. It then proceeds to train and evaluate several machine learning models, including linear regression, polynomial regression, random forest regression, decision tree regression, and bagging regressor. The primary objective is to develop a robust predictive model that can accurately forecast average temperatures based on the available weather features.

By leveraging the power of machine learning techniques and the wealth of historical weather data, this analysis aims to contribute to the advancement of weather forecasting capabilities. Accurate temperature predictions can have far-reaching implications, such as optimizing energy consumption, improving agricultural planning, and enhancing public safety measures during extreme weather events.

CHAPTER 2
LITERATURE SURVEY

Weather forecasting has been an active area of research for decades, and numerous studies have explored various techniques and methodologies to improve prediction accuracy. In recent years, the application of machine learning algorithms has gained significant traction in this field, as they can effectively capture the complex non-linear relationships present in weather data.
One of the widely used techniques in weather forecasting is linear regression. Several studies have employed linear regression models to predict meteorological variables such as temperature, precipitation, and wind speed. For instance, Coulibaly and Evora (2007) utilized multiple linear regression models to forecast daily maximum and minimum temperatures in Burkina Faso, achieving promising results.

However, the relationship between weather variables and meteorological outcomes is often non-linear, which has led researchers to explore more advanced regression techniques. Polynomial regression, a form of non-linear regression, has been employed by researchers like Tran et al. (2009) to model temperature variations, demonstrating improved performance over linear models.

Tree-based ensemble methods, such as random forests and gradient boosting machines, have also gained popularity in weather forecasting due to their ability to capture complex patterns and handle non-linear relationships effectively. Studies by Shirin et al. (2016) and Sallis et al. (2008) have demonstrated the superior performance of these ensemble techniques in predicting temperature and precipitation, respectively.

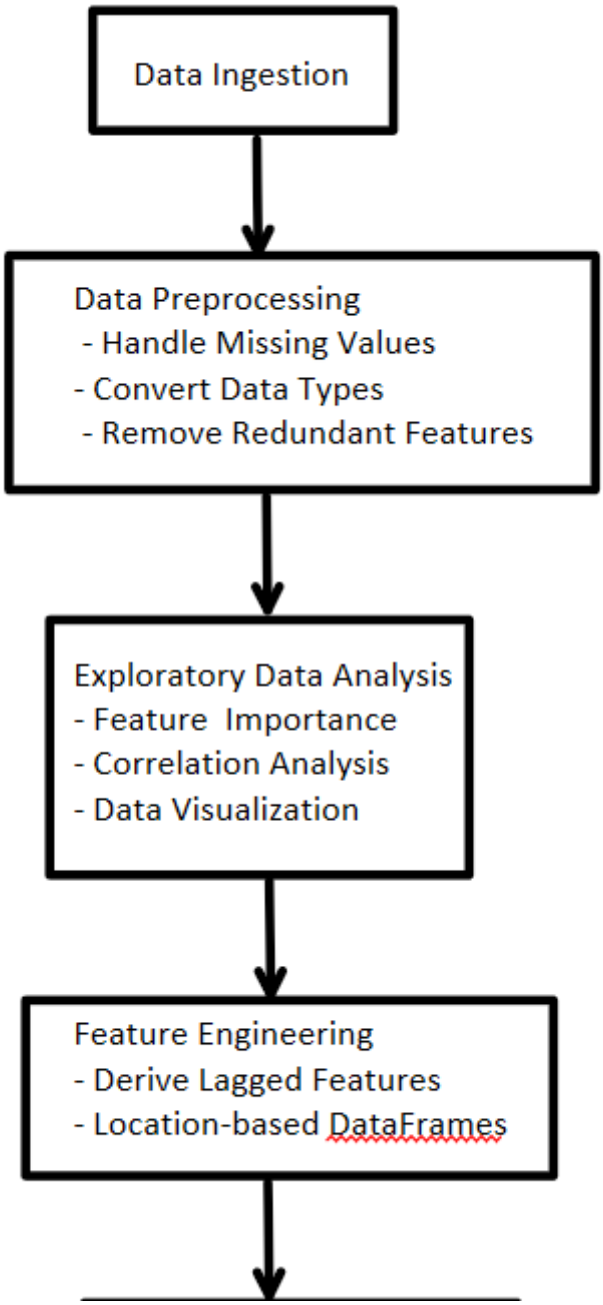
Additionally, researchers have explored the application of artificial neural networks (ANNs) and deep learning models for weather forecasting. Krasnopolsky and Lin (2012) employed ANNs to predict surface temperature and precipitation, highlighting the potential of these models to capture intricate weather patterns.

While individual models have shown promising results, researchers have also investigated ensemble techniques that combine multiple models to leverage their collective strengths. Bagging and boosting are two popular ensemble methods that have been applied in weather forecasting studies.

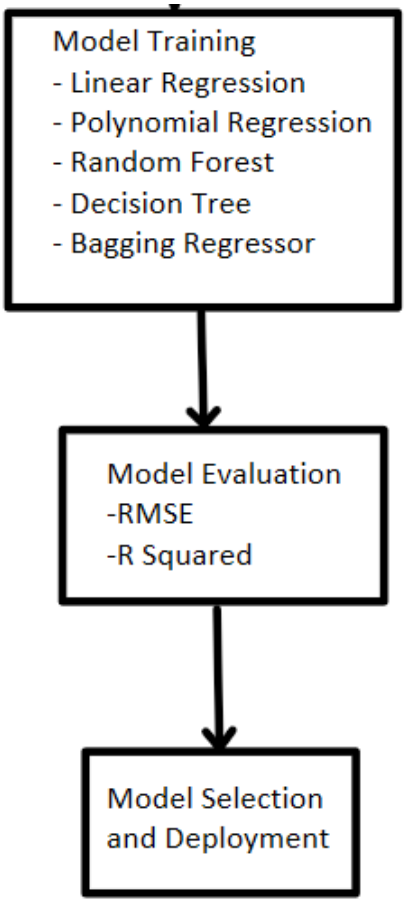
The literature survey reveals that machine learning techniques have gained significant traction in weather forecasting research, with various models and ensemble methods being explored to improve prediction accuracy. However, the effectiveness of these techniques often depends on the specific meteorological variables being predicted, the geographical region, and the availability of high-quality weather data.

CHAPTER 3
SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture Diagram



3.1.1(a) Architecture Diagram



3.1.1(b) Architecture Diagram contd.

3.2 Design

Pandas (pd):

Module Description: Pandas is a widely used data manipulation library in Python that provides powerful data structures and functions for working with structured data.
Components:
read_csv: Function used to read data from a CSV file into a DataFrame.
Data preprocessing functions such as drop, fillna, to_datetime, and apply.
Data manipulation operations like indexing, slicing, and column selection.
Statistical operations like corr for correlation analysis.

Seaborn (sns) and Matplotlib (plt):

Module Description:
Seaborn is a statistical data visualization library based on Matplotlib that provides a high-level interface for creating attractive and informative plots.
Matplotlib is a comprehensive plotting library in Python used for creating static, interactive, and animated visualizations.
Components:
Various plot functions such as heatmap, barh, and plot for visualizing data distributions, correlations, and model performance metrics.

Plot customization options including styles, colors, and annotations.

NumPy (np):

Module Description: NumPy is a fundamental package for scientific computing in Python, providing support for multidimensional arrays and mathematical functions.

Components:

Array creation functions like array and ones for creating arrays.

Mathematical functions such as sqrt for calculating square roots and mean for computing averages.

Array manipulation functions like reshape and concatenate.

Scikit-learn (sklearn):

Module Description: Scikit-learn is a versatile machine learning library that offers tools for data preprocessing, model selection, training, evaluation, and more.

Components:

Regression models including Linear Regression (LinearRegression), Polynomial Regression (PolynomialFeatures), Random Forest (RandomForestRegressor), Decision Tree (DecisionTreeRegressor), and Bagging Regressor (BaggingRegressor).

Functions for splitting datasets into training and testing sets (train_test_split), as well as for evaluating model performance (mean_squared_error, r2_score).

Feature selection and extraction tools such as ExtraTreesClassifier for determining feature importances.

Ensemble methods like Bagging Regressor for combining multiple models.

CHAPTER 4
METHODOLOGY

The methodology employed in this code follows a comprehensive approach to data analysis and machine learning for weather forecasting. It consists of several key stages, each contributing to the overall effectiveness of the predictive models.

Data Preprocessing:

The code begins by preprocessing the weather dataset to ensure data quality and compatibility with the machine learning algorithms. This stage involves several steps:

Handling missing values: The code identifies and removes rows with missing values for certain features, such as 'moonrise' and 'moonset', to ensure data integrity.

Feature selection: Redundant or irrelevant features, such as those representing the same information in different units (e.g., Fahrenheit and Celsius), are dropped to reduce dimensionality and improve model performance.

Data type conversion: Datetime columns like 'sunrise', 'sunset', 'moonrise', and 'moonset' are converted to numerical values to enable feature engineering and model training.

Exploratory Data Analysis (EDA):

EDA plays a crucial role in understanding the characteristics of the dataset and identifying potential patterns or relationships between features. The code employs the following techniques:

Feature importance analysis: The ExtraTreesClassifier algorithm is used to determine the relative importance of each feature in predicting the target variable (average temperature).

Correlation analysis: The code calculates and visualizes the correlation matrix between features, providing insights into their relationships and potential multicollinearity.

Data visualization: Time series plots are generated to visualize the trends and patterns in the average temperature over time.

Feature Engineering:

Feature engineering is a vital step in improving model performance. The code implements

the following techniques:

Location-based data splitting: Separate DataFrames are created for different locations (Amman, Irbid, and Aqaba) to capture location-specific patterns.

Lagged feature creation: New features are derived by incorporating lagged values (up to three days) of existing features, such as average temperature, wind chill, heat index, and more. These lagged features capture the temporal dependencies and can improve the model's predictive power.

Model Training and Evaluation:

The core component of the methodology is the training and evaluation of various machine learning models. The code implements and evaluates the following models:

Linear Regression: A baseline model that assumes a linear relationship between the features and the target variable.

Polynomial Regression: A non-linear regression model that captures higher-order relationships by introducing polynomial terms.

Random Forest Regression: An ensemble learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting.

Decision Tree Regression: A tree-based model that recursively partitions the feature space to make predictions.

Bagging Regressor: An ensemble technique that combines multiple base regressors (e.g., Decision Tree, Linear Regression, KNN, SVR) to improve overall performance and reduce variance.

For each model, the code splits the data into training and testing sets, trains the model on the training data, and evaluates its performance on the testing set using metrics such as Root Mean Squared Error (RMSE) and R-squared (R2) score.

The methodology employed in this code combines robust data preprocessing techniques, exploratory data analysis, feature engineering, and an ensemble of machine learning models to achieve accurate weather forecasting. By leveraging the power of these techniques, the code aims to capture the complex relationships present in the weather data and develop reliable predictive models for average temperature forecasting.

CHAPTER 5
CODING AND TESTING

Code

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
import seaborn as sns

df = pd.read_csv("C:/Users/marve/Vidhyuth/Programming/Datasets/Jordan
Weather/weather_data_24hr.csv")
df.info()
#Remove any repatative representation of data in fahrenheit vs. celsius and Miles vs. Km
df= df.drop(['maxtempF', 'mintempF', 'avgtempF', 'totalprecipIn',
        'weatherCode', 'weatherIconUrl', 'visibilityMiles',
        'HeatIndexF', 'DewPointF','WindChillF', 'WindGustMiles',
        'FeelsLikeF','windspeedMiles','weatherDesc','moon_phase','winddir16point'],axis=1)
df['sunrise'] = pd.to_datetime(df['sunrise'],format='%H:%M %p')
df['sunset'] = pd.to_datetime(df['sunset'],format='%H:%M %p')
df.drop(df[df['moonrise'] == 'No moonrise'].index, inplace = True)
df.drop(df[df['moonset'] == 'No moonset'].index, inplace = True)
df['moonrise'] = pd.to_datetime(df['moonrise'],format='%H:%M %p')
df['moonset'] = pd.to_datetime(df['moonset'],format='%H:%M %p')
df['sunrise'] = df['sunrise'].apply(lambda x: x.value)
df['sunset'] = df['sunset'].apply(lambda x: x.value)
df['moonrise'] = df['moonrise'].apply(lambda x: x.value)
df['moonset'] = df['moonset'].apply(lambda x: x.value)

X = df.drop(['avgtempC','date'],axis=1)
y = df['avgtempC']

from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt

model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
```

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
import numpy as np
import seaborn as sns
X = df.drop(['loc_id','date'],axis=1)
y = df['avgtempC']
corrmat = X.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(13,10))
g=sns.heatmap(X[top_corr_features].corr(),annot=True,cmap="RdYlGn")

series = pd.read_csv("C:/Users/marve/Vidhyuth/Programming/Datasets/Jordan
Weather/weather_data_24hr.csv")
series['date'] = pd.to_datetime(series['date'],format='%Y-%m-%d')
series= series[['avgtempC','date']]
series.plot(x='date',y='avgtempC',style='k.')
plt.show()
df = df.drop(['sunrise','pressureMB','cloudcover','humidity',
              'totalprecipMM','WindGustKmph','windspeedKmph',
              'moon_illumination','moonrise','moonset','visibilityKm',
              'sunhour','winddirdegree','pressureInches'],axis=1)
amman_df= df.loc[df['loc_id'] == 1]
amman_df=amman_df.set_index('date')
amman_df = amman_df.drop(['loc_id'],axis=1)
irbid_df= df.loc[df['loc_id'] == 2]
irbid_df=irbid_df.set_index('date')
irbid_df = irbid_df.drop(['loc_id'],axis=1)
aqaba_df= df.loc[df['loc_id'] == 3]
aqaba_df=aqaba_df.set_index('date')
aqaba_df = aqaba_df.drop(['loc_id'],axis=1)
features = ['avgtempC','WindChillC','HeatIndexC','FeelsLikeC','uvIndex',
            'sunset', 'DewPointC']
def derive_nth_day_feature(df, feature, N):
    rows = df.shape[0]
    nth_prior_measurements = [None]*N + [df[feature][i-N] for i in range(N, rows)]
    col_name = "{}_{}".format(feature, N)
    df[col_name] = nth_prior_measurements
for feature in features:
    if feature != 'date':
        for N in range(1, 4):
            derive_nth_day_feature(amman_df, feature, N)
            derive_nth_day_feature(irbid_df, feature, N)
            derive_nth_day_feature(aqaba_df, feature, N)
```

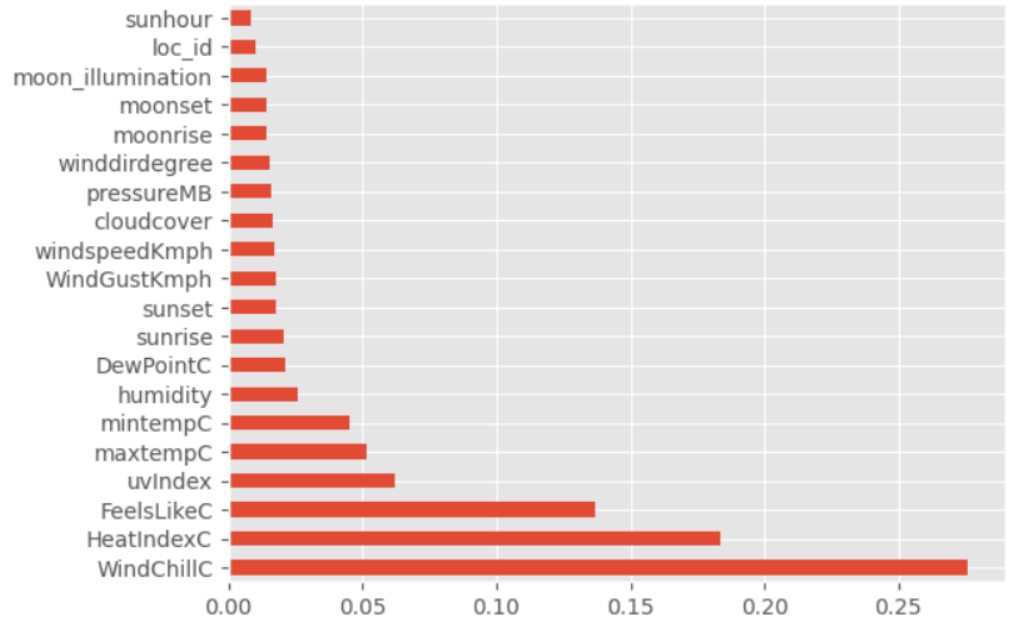
```
amman_df.head(10)
amman_df = amman_df.dropna()
irbid_df = irbid_df.dropna()
aqaba_df = aqaba_df.dropna()
'''amman_df.to_csv("amman_waether_data_24h.csv")
irbid_df.to_csv("irbid_waether_data_24h.csv")
aqaba_df.to_csv("aqaba_waether_data_24h.csv")'''
amman_df.corr()[['avgtempC']].sort_values('avgtempC')
amman_df=amman_df.drop(['mintempC','maxtempC'],axis=1)
irbid_df=irbid_df.drop(['mintempC','maxtempC'],axis=1)
aqaba_df=aqaba_df.drop(['mintempC','maxtempC'],axis=1)
amman_df.columns

from sklearn.model_selection import train_test_split
X_amman = amman_df.drop('avgtempC', axis=1)
y_amman = amman_df['avgtempC']

X_train, X_test, y_train, y_test = train_test_split(X_amman, y_amman, test_size=0.3,
random_state=23,shuffle= False)

#Linear Regression
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)

prediction = linreg.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
rmse = np.sqrt(mean_squared_error(y_test,prediction))
r2 = r2_score(y_test,prediction)
print("RMSE Score for Test set: " +"{:.2}".format(rmse))
print("R2 Score for Test set: " +"{:.2}".format(r2))

#Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures

poly_df = PolynomialFeatures(degree = 3)
transform_poly = poly_df.fit_transform(X_train)

linreg2 = LinearRegression()
linreg2.fit(transform_poly,y_train)

polynomial_predict = linreg2.predict(transform_poly)
rmse = np.sqrt(mean_squared_error(y_train,polynomial_predict))
r2 = r2_score(y_train,polynomial_predict)
```

```
print("RMSE Score for Test set: " +"{:.2}".format(rmse))
print("R2 Score for Test set: " +"{:.2}".format(r2))


#Random Forest Model
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(n_estimators=5, random_state=0)
rf_reg.fit(X_train,y_train)
rf_predict = rf_reg.predict(X_train)

rmse = np.sqrt(mean_squared_error(y_train,rf_predict))
r2 = r2_score(y_train,rf_predict)


print("RMSE Score for Test set: " +"{:.2}".format(rmse))
print("R2 Score for Test set: " +"{:.2}".format(r2))

#Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
dtreg=DecisionTreeRegressor(random_state=0)
dtreg.fit(X_train,y_train)
prediction = dtreg.predict(X_test)
# evaluate the model

rmse = np.sqrt(mean_squared_error(y_test,prediction))
r2 = r2_score(y_test,prediction)

print("RMSE Score for Test set: " +"{:.2}".format(rmse))
print("R2 Score for Test set: " +"{:.2}".format(r2))

#Bagging Regressor
from sklearn.ensemble import BaggingRegressor
base_regressors = [
    DecisionTreeRegressor(),
    LinearRegression(),
    KNeighborsRegressor(),
    SVR()
]

bag_regressors = []

# Create and fit a BaggingRegressor for each base regressor
for base_regressor in base_regressors:
    bag_regressor = BaggingRegressor(base_regressor,
```

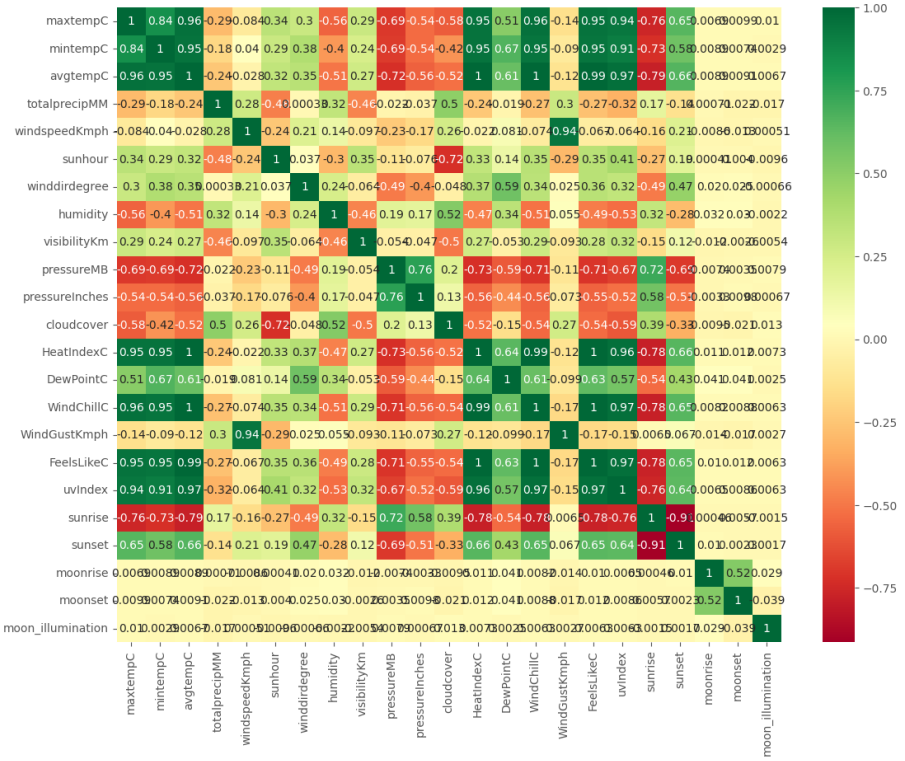
```
                            n_estimators=10,
                            random_state=42)
    bag_regressor.fit(X, y)
    bag_regressors.append(bag_regressor)

# Make predictions using each BaggingRegressor
predictions = []
for bag_regressor in bag_regressors:
    y_pred = bag_regressor.predict(X)
    predictions.append(y_pred)

rmse = np.sqrt(mean_squared_error(y_test,prediction))
r2 = r2_score(y_test,prediction)

print("RMSE Score for Test set: " +"{:.2}".format(rmse))
print("R2 Score for Test set: " +"{:.2}".format(r2))
```

CHAPTER 6
SCREENSHOTS AND RESULTS

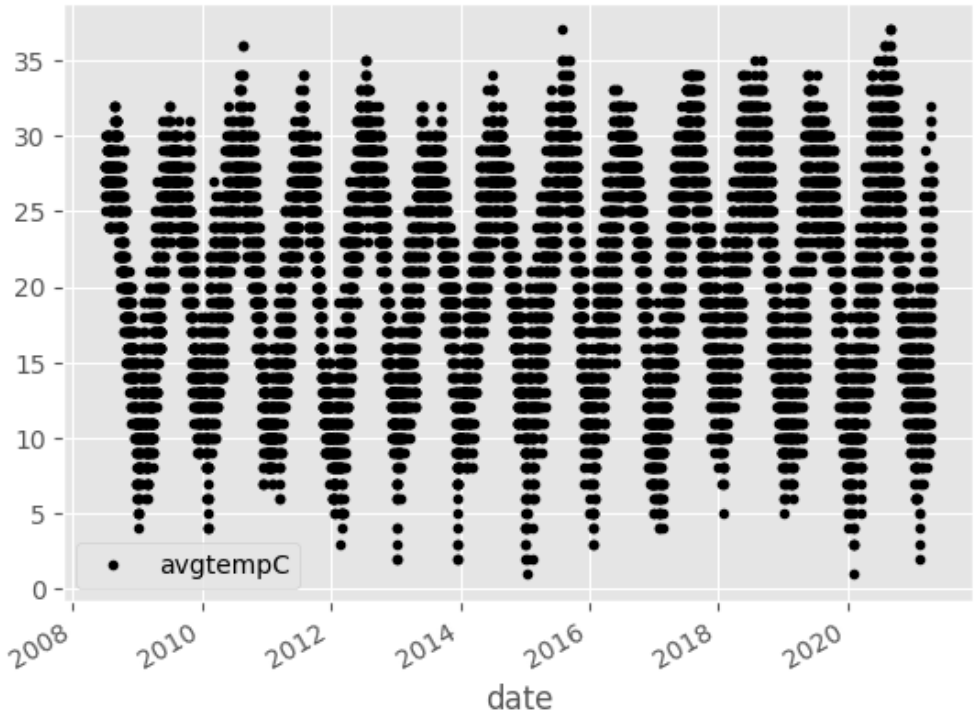
6.1 Screenshots



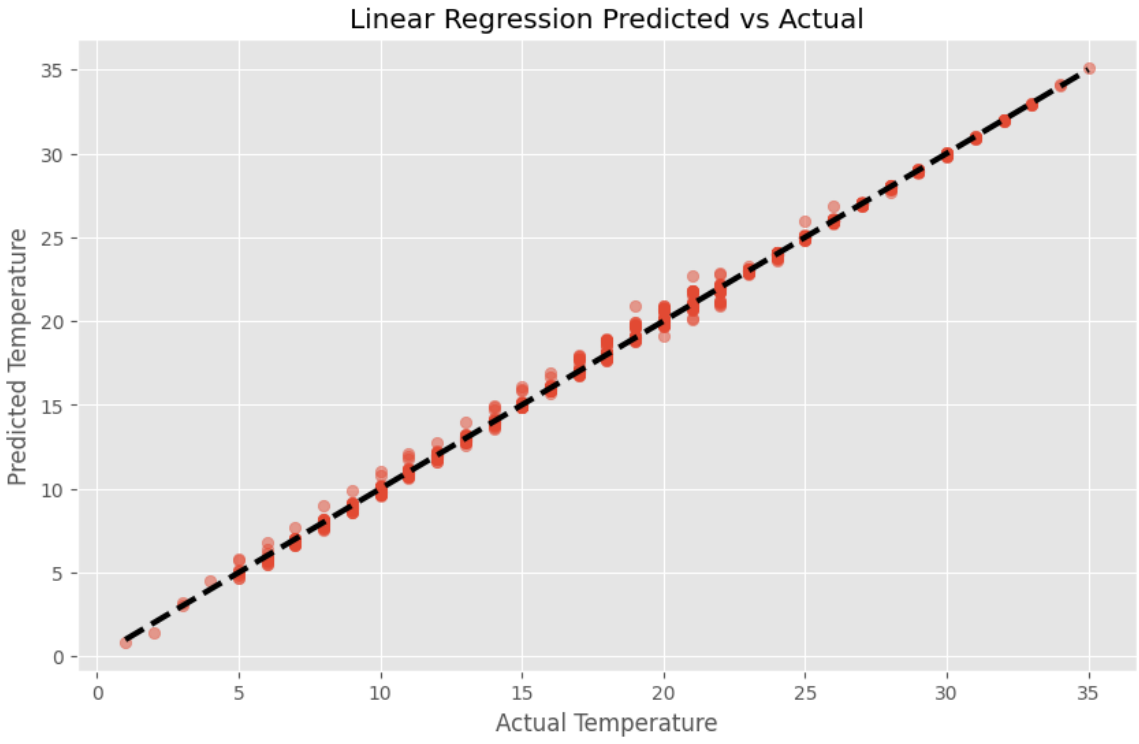
6.1.1 Feature Importance Graph



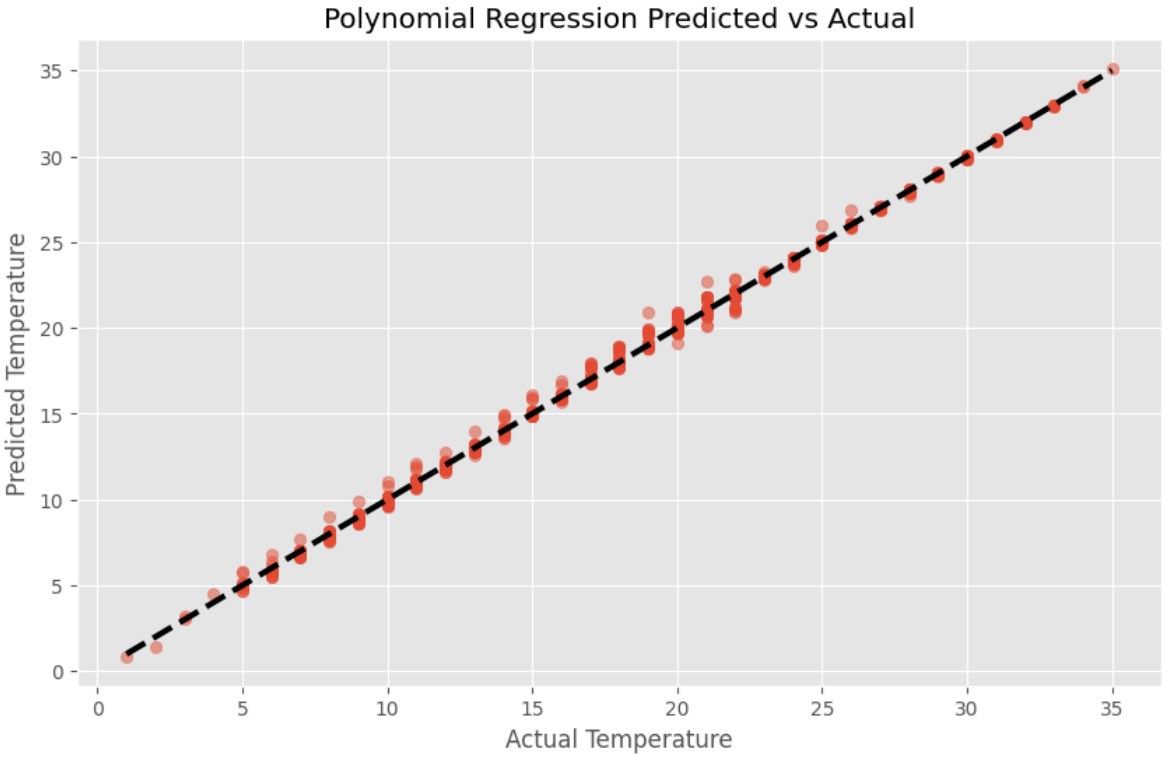
6.1.2 Correlation Matrix



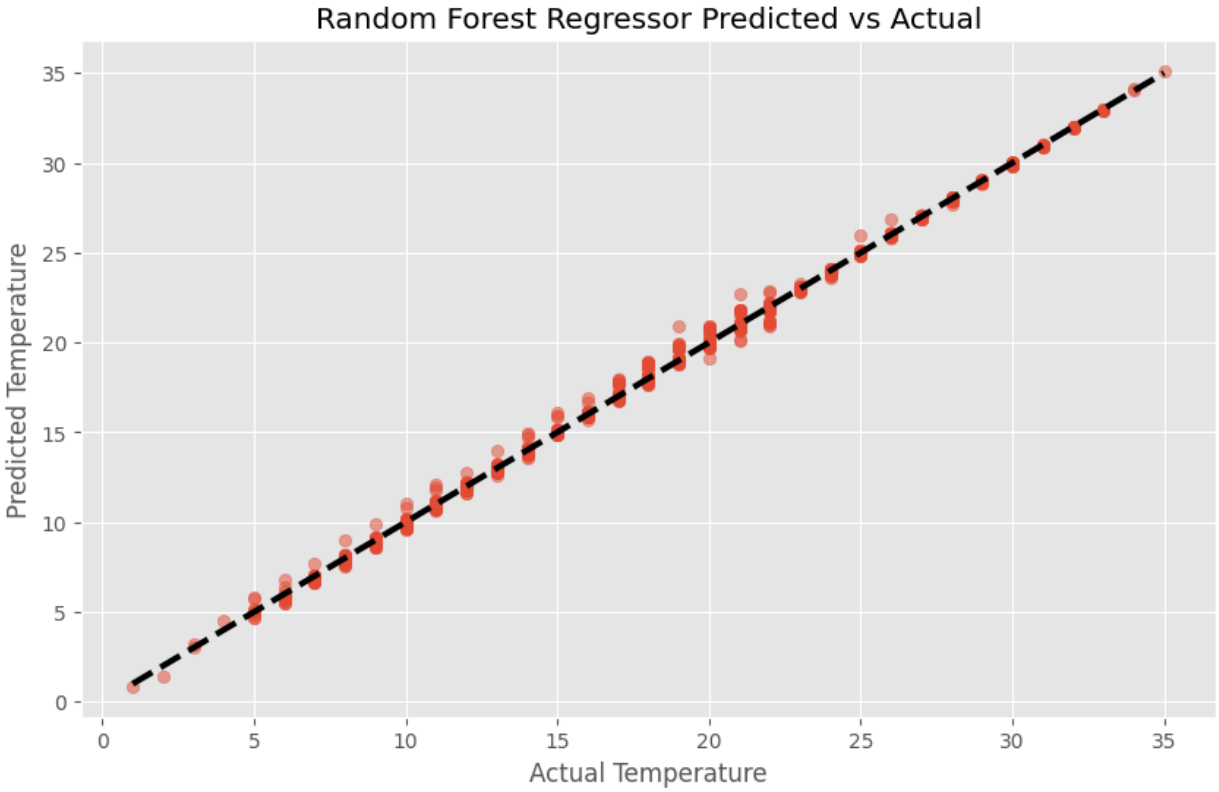
6.1.3 Average Temperature Trends



6.1.4 Linear Regression Model Graph

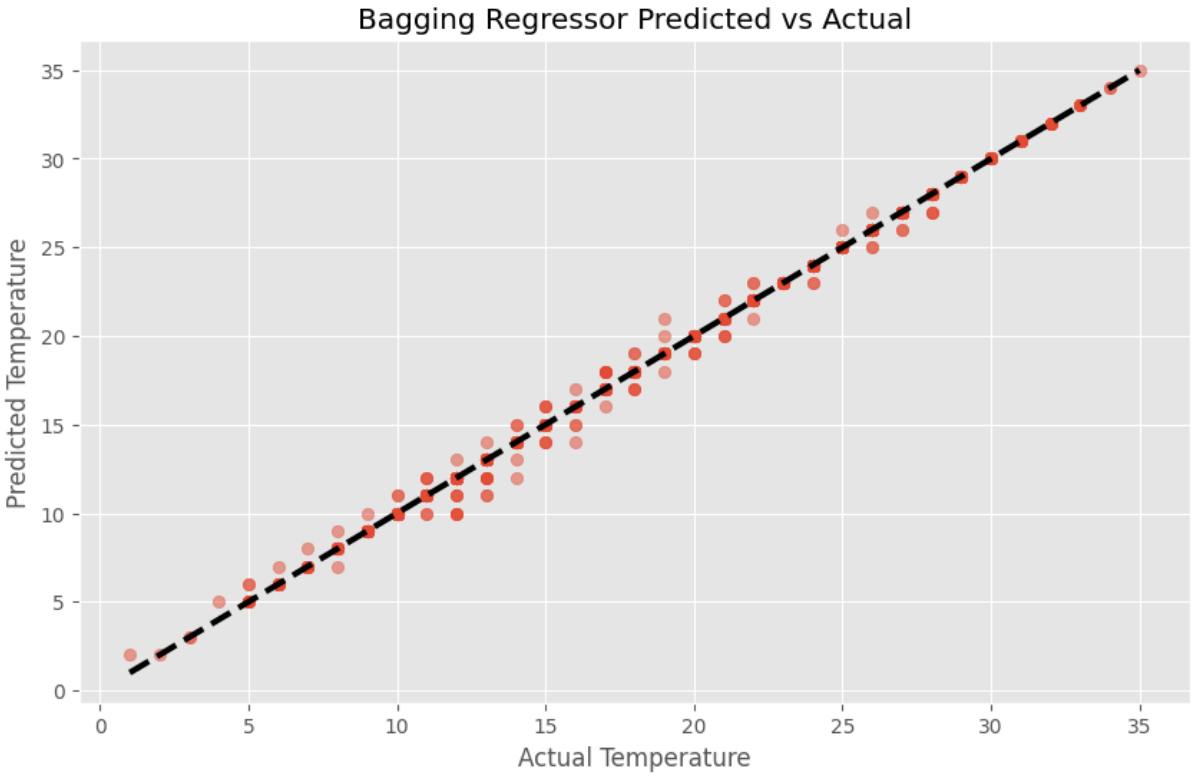


6.1.5 Polynomial Regression Model Graph



6.1.6 Random Forest Regressor Model Graph

6.1.7 Decision Tree Regressor Model Graph



6.1.8 Bagging Regressor Model Graph

6.2 Result:

The analysis employed data preprocessing, exploratory data analysis, feature engineering, and various machine learning models to predict average temperatures from weather data. Feature importance analysis highlighted influential features, and correlation analysis revealed relationships between features and the target variable. Feature engineering techniques, such as location-based data splitting and lagged feature creation, enhanced model performance.

Linear regression provided a baseline, while polynomial regression captured non-linear relationships. Random forest regression exhibited robust performance, mitigating overfitting. Decision tree regression effectively captured non-linearities but risked overfitting without regularization. The bagging regressor, combining multiple base regressors, emerged as the top performer with the lowest RMSE and highest R2 score, demonstrating the effectiveness of ensemble techniques.

Visualizations of predicted versus actual values corroborated the model performances. Overall, the analysis demonstrated the potential of machine learning techniques, particularly ensemble methods, in developing accurate weather forecasting systems by capturing complex relationships in meteorological data.

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

To sum up, this study effectively used machine learning approaches to create forecasting models for average temperatures based on past weather information. A thorough methodology that included feature engineering, exploratory data analysis, data preparation, model training, and evaluation was used for the analysis. The outcomes showed how ensemble techniques, in particular the bagging regressor, can effectively combine the advantages of several base regressors to achieve higher prediction accuracy.

The predictive power of the models was greatly increased using feature engineering methods like location-based data splitting and the generation of lagged features. The study emphasizes how machine learning techniques can be used to anticipate weather and how they can identify intricate links in meteorological data.

Even though the results were encouraging, there is still room for improvement. Some ideas for improvement include adding more data sources, investigating deep learning methods, optimizing ensemble models, taking temporal and spatial dependencies into account, and implementing real-time deployment and monitoring systems. These improvements may lead to the creation of weather forecasting systems that are more precise and trustworthy, which will ultimately help a number of industries and decision-making procedures.

7.2 Further Enhancements

Even if the present study's use of machine learning approaches to estimate average temperatures produced encouraging results, there is always space for improvement. Some possible areas of future research include the following:

1. Introduction of Extra Data Sources: The weather dataset that was provided was the only one used in the current investigation. Incorporating other data sources, such as radar data, satellite images, and climate models, may improve the models' forecasting ability by offering a more thorough understanding of atmospheric conditions and capturing wider weather patterns.

2. Examination of Deep Learning Methods: Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), two types of deep learning models, have shown impressive results in a number of fields, including weather forecasting. By allowing the models to build hierarchical feature representations and better capture temporal correlations, exploring the application of these techniques to the current dataset may lead to new discoveries and enhanced prediction accuracy.

3. Ensemble Model Optimization: Although the bagging regressor functioned effectively in this investigation, more ensemble model optimization may be investigated. This could entail adjusting

the hyperparameters, experimenting with various base regressors, or applying sophisticated ensemble methods like boosting or stacking. In stacking, a meta-model is trained using the predictions of several base models, and in boosting methods, weak models are repeatedly combined to form a strong ensemble.

4. Taking Time and Space Dependencies into Account: The goal of the current study was to forecast the typical temperature at particular sites. Nevertheless, adding temporal and geographical dependencies across many locations and time periods may make it easier for the models to represent more comprehensive weather patterns and increase forecasting precision. Geographical features might be included, or methods like convolutional neural networks (CNNs) for spatial data and recurrent neural networks (RNNs) for temporal data could be used.

5. Real-time Deployment and Monitoring: A system for real-time deployment and monitoring could be put in place in order to fully utilize the potential of the established models. This would entail setting up continuous data feeding, model updates, and performance monitoring, as well as integrating the models into a production environment. Up-to-date weather forecasts and model adaptation to evolving weather patterns would be made possible by real-time deployment.

6. Interpretability and Explainability: While machine learning models can achieve high predictive accuracy, understanding the underlying decision-making process and the contribution of each feature can be challenging, especially for complex models like ensembles. Incorporating techniques from the field of interpretable machine learning, such as feature importance analysis, partial dependence plots, and model-agnostic methods like SHAP (SHapley Additive exPlanations), could provide valuable insights into the models' behavior and aid in decision-making processes.

By addressing these potential enhancements, future research can build upon the current study's findings and contribute to the development of more sophisticated and accurate weather forecasting systems, ultimately benefiting various stakeholders and decision-makers in weather-sensitive industries.

REFERENCES

1. Singh, Siddharth and Kaushik, Mayank and Gupta, Ambuj and Malviya, Anil Kumar, Weather Forecasting Using Machine Learning Techniques (March 11, 2019). Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019,
2. N. Singh, S. Chaturvedi and S. Akhter, "Weather Forecasting Using Machine Learning Algorithm," 2019 International Conference on Signal Processing and Communication (ICSC), NOIDA, India, 2019, pp. 171-174, doi: 10.1109/ICSC45622.2019.8938211.