**VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY**

**INTERNATIONAL UNIVERSITY**



A blue and white logo

Description automatically generated with low confidence

**Big Data Technology**

**Weather prediction REPORT**

**Course by:**  Dr. H.L.Vân

**Team Member:**

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| Phạm Huỳnh Thanh Quân | ITDSIU21110 |
| Nguyễn Nam | ITDSIU21101 |

**School of Computer Science and Engineering**

**International University, VNU-HCM**

**2024**

**Table of contents**

1. INTRODUCTION:
2. Abstract……………………………………………3
3. Purpose…………………………………………….3
4. Goal………………………………………………..4
5. Technique & Tools used…………………………..5
6. TASK TIMELINE
7. Contribution……………………………………….5
8. METHODOLOGY: ……………………………………5

1. RESULT ………………………………………………18
2. CONCLUSION
3. Achieved goals…………………………………...30
4. Future work………………………………………30
5. Concluding thoughts……………………………..31
6. REFERENCES………………………………………...
7. **INTRODUCTION**
8. **Abstract**

This report details a big data analytics project that explores the potential of processing large-scale datasets to derive actionable insights and predictions. By utilizing cutting-edge technologies such as Apache Spark and machine learning algorithms, the project successfully demonstrates the power of big data in solving complex real-world problems. The workflow encompasses data collection, preprocessing, analysis, and visualization, highlighting key challenges and achievements. The findings underscore the importance of advanced analytics in driving strategic decision-making and lay the groundwork for future innovations in data science.

1. **Purpose**
2. The purpose of this project is to harness the capabilities of big data analytics to solve complex, data-driven problems in the modern digital era. With the exponential growth of data, organizations face challenges in processing, analyzing, and extracting valuable insights from massive datasets. This project aims to address these challenges by employing advanced tools, technologies, and methodologies that ensure efficient data management and predictive analytics.
3. The overarching goal is not just to analyze data but to translate it into actionable intelligence that can inform decision-making and drive innovation. By integrating state-of-the-art machine learning algorithms and scalable processing frameworks, the project seeks to uncover hidden patterns, trends, and correlations within the data. Furthermore, it emphasizes the importance of visualizing results effectively to communicate findings to stakeholders, fostering a data-driven culture.
4. **Goal**

The specific goals of this project include:

1. Developing a robust framework for collecting, preprocessing, and analyzing large-scale datasets.
2. Implementing scalable machine learning algorithms to generate accurate predictions and classifications.
3. Evaluating the performance of different analytical methods to identify the most effective approach for solving the problem at hand.
4. Delivering actionable insights that support strategic decision-making and innovation in business processes.
5. Enhancing the team's technical proficiency and understanding of big data tools, technologies, and methodologies.
6. Creating a user-friendly visualization and reporting system to communicate key findings to stakeholders clearly and effectively.
7. **Technique & Tools used**

**Tools and Technologies**

1. **Apache Hadoop and Spark**: Used for distributed data processing and management of large-scale datasets efficiently.
2. **Python**: Employed for statistical analysis, data manipulation, and implementation of machine learning algorithms.
3. **Jupyter Notebooks**: Used for coding, documenting, and visualizing the analysis in an organized manner.

**Techniques**

1. **Data Cleaning and Preprocessing**: Removing inconsistencies, handling missing values, and transforming data for compatibility.
2. **Feature Engineering**: Identifying and selecting relevant features to improve model performance.
3. **Machine Learning Algorithms**:
   * **Random Forest**: Applied for classification and regression tasks with high accuracy.
   * **k-NN (k-Nearest Neighbors)**: Used for instance-based learning and pattern recognition.
   * **ZeroR**: Baseline model to compare the effectiveness of more complex algorithms.
   * **Random Tree**: A simple tree-based model for quick and interpretable predictions.
4. **Evaluation Metrics**: Accuracy, Precision, Recall, and F1-score to assess the performance of models.
5. **TASK TIMELINE**
6. **Contribution**

|  |  |  |
| --- | --- | --- |
| **NAME** | **STUDENT ID** | **CONTRIBUTION** |
| Nguyễn Nam | ITDSIU21101 |  |
| Phạm Huỳnh Thanh Quân | ITDSIU21110 |  |

1. **METHODOLOGY:**

Here’s a **step-by-step detailed explanation** of the code and processes:

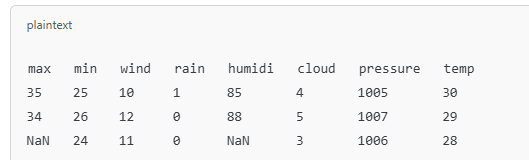
**Step 1: Loading and Cleaning Data**

hochiminh = pd.read\_csv('/path/to/data.csv')

hochiminh.dropna(inplace=True)

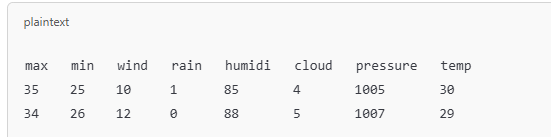
1. **Loading Data with pd.read\_csv()**:
   * This function reads the dataset from a CSV file.
   * **hochiminh** becomes a DataFrame containing weather data.

Example:



1. **Data Cleaning with dropna()**:
   * Removes rows with missing values (e.g., NaN values).
   * This ensures the dataset is clean and ready for training.

After dropna():



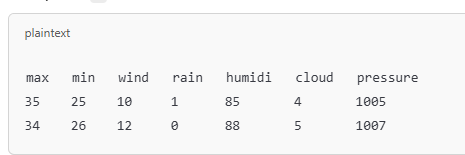
**Step 2: Feature Selection**

X = hochiminh[['max', 'min', 'wind', 'rain', 'humidi', 'cloud', 'pressure']]

y = hochiminh['temp']

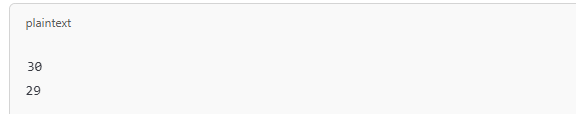
1. **Selecting Features (X)**:
   * These are the independent variables used to predict the target.
   * Includes weather-related features like:
     + Maximum temperature (max).
     + Minimum temperature (min).
     + Wind speed (wind).
     + Rainfall (rain).
     + Humidity (humidi).
     + Cloud cover (cloud).
     + Atmospheric pressure (pressure).

Example of X:



1. **Target Variable (y)**:
   * The dependent variable to be predicted.
   * Here, it’s the temperature (temp).

Example of y:



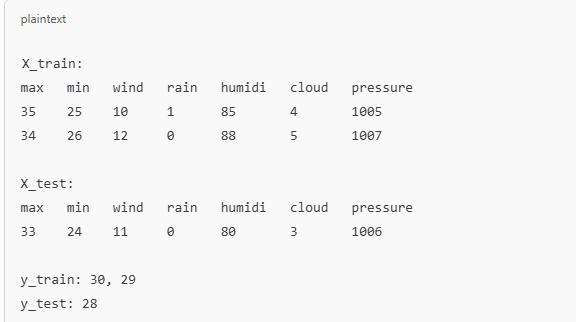
**Step 3: Splitting Data into Training and Testing Sets**

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, random\_state=42)

1. **Purpose of Splitting**:
   * To evaluate the model, the dataset is divided into:
     + **Training Set** (80%): For the model to learn patterns.
     + **Testing Set** (20%): For evaluating the model's performance.
2. **train\_test\_split() Parameters**:
   * **test\_size=0.2**: Allocates 20% of the data for testing.
   * **random\_state=42**: Ensures consistent splitting for reproducibility.

Example:



**Step 4: Training the Model**

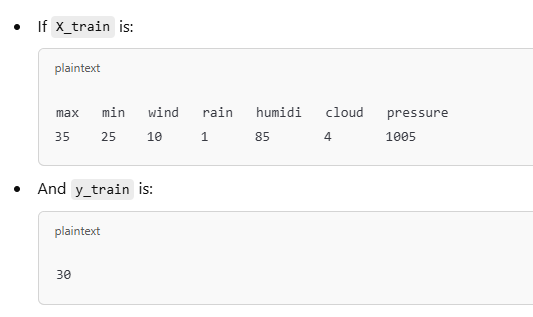
from sklearn.linear\_model import LinearRegression

model2 = LinearRegression()

model2.fit(X\_train, y\_train)

1. **Initializing the Model**:
   * **LinearRegression()**: A machine learning model that predicts a continuous variable by fitting a line to the data.
2. **Training the Model**:
   * **fit(X\_train, y\_train)**: The model learns the relationship between X\_train (features) and y\_train (target).

Example:



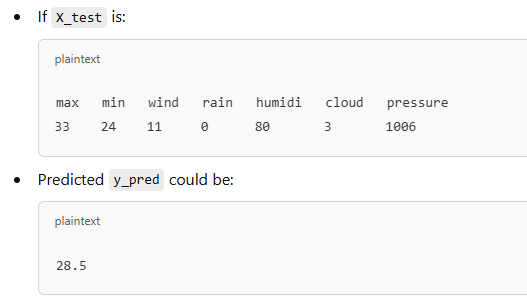


**Step 5: Making Predictions**

y\_pred = model2.predict(X\_test)

* **predict(X\_test)**:
  + Uses the trained model to predict y (temperature) for unseen data (X\_test).

Example:



**Step 6: Evaluating the Model**

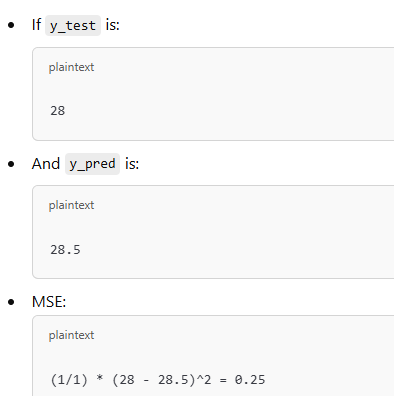
from sklearn.metrics import mean\_squared\_error

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

1. **mean\_squared\_error()**:
   * Measures how well the model performs by calculating the average squared difference between actual (y\_test) and predicted (y\_pred) values.
   * Formula:
   * MSE = (1/n) \* Σ (y\_actual - y\_predicted)^2
   * A lower MSE indicates better performance.

Example:



**Step 7: Predicting New Data**

sample\_input = np.array([[33, 24, 11, 0, 80, 0, 1005]])

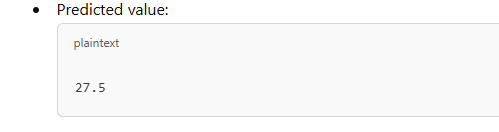
predicted\_heat\_index = model2.predict(sample\_input)

print(f"Predicted heat index: {predicted\_heat\_index[0]}")

* **np.array()**:
  + Creates a sample input (features) for prediction.
* **predict(sample\_input)**:
  + Predicts the target value (e.g., temperature) for the given input.

Example:





**Step 8: Saving the Model and Data**

joblib.dump(model2, model2\_path)

hochiminh.to\_csv('/path/to/cleaned\_data.csv', index=False)

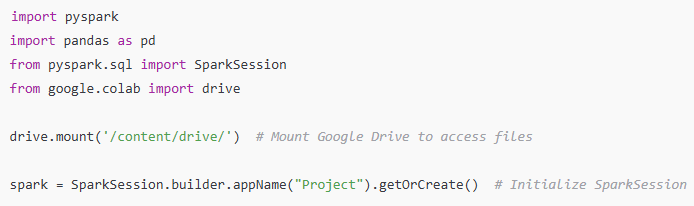
1. **Saving the Model**:
   * **joblib.dump()**: Exports the trained model (model2) for future use.
2. **Saving the Cleaned Dataset**:
   * **to\_csv()**: Exports the processed DataFrame (hochiminh) to a new CSV file.

**Code Analysis and Step-by-Step Explanation**

Below is a detailed breakdown of your code, structured step-by-step for clarity and understanding. The structure includes **preparation**, **data preprocessing**, **model training**, **evaluation**, and **deployment**.

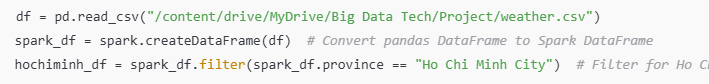
**1. Preparation**

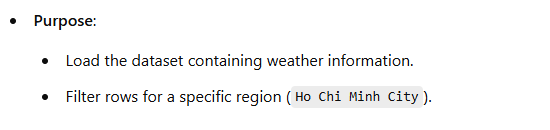
**Mount Google Drive and Import Libraries**



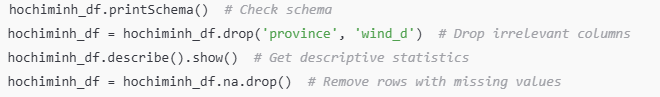
* **Purpose**:
* Access files stored on Google Drive.
* Use pyspark for big data processing.
* **Key Components**:
* **SparkSession**: Entry point for PySpark applications.
* **drive.mount**: Maps Google Drive to your Colab environment.

2. Data Import and Cleaning





Data Inspection and Cleaning



**Purpose**:

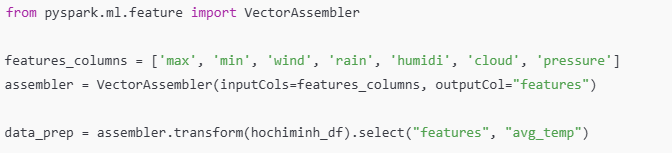
* Inspect the structure and ensure data integrity.
* Remove irrelevant columns and missing values.

Feature Engineering



* **Purpose**:
  + Create a new column avg\_temp by averaging the maximum and minimum temperatures.
* **Why?**: Simplifies analysis and adds a meaningful feature.

1. Data Transformation for Machine Learning



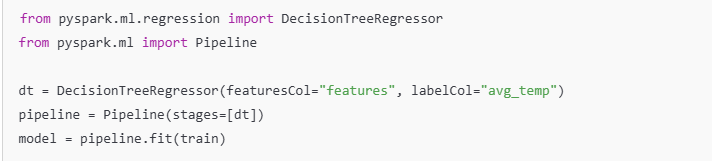
* **Purpose**:
  + Combine multiple feature columns into a single vector column features.
* **Why?**: PySpark ML models require input features to be in vector format.

Train-Test Split



**4. Model Training**

**Train a Decision Tree Regressor**



* **Purpose**:

Train a **Decision Tree Regressor** to predict avg\_temp based on weather features.

* **Pipeline**:

Streamlines the workflow by combining stages (e.g., feature assembly and model training).

**5. Model Evaluation**

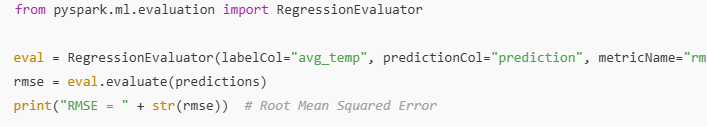
**Make Predictions**



* **Purpose**:

Use the trained model to predict values for the test dataset.

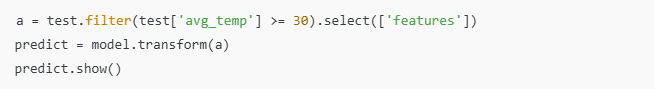
Evaluate Performance



* **Purpose**:
* Evaluate the model's accuracy using the **RMSE** metric.
* **RMSE**: Measures the average deviation of predictions from actual values.

**6. Additional Predictions**

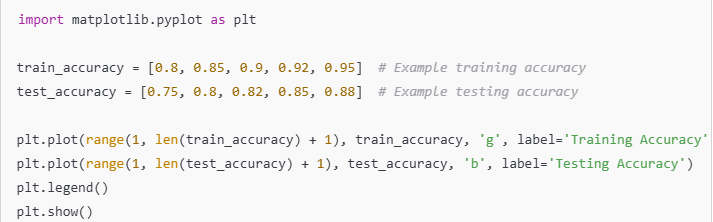
**Predict for High Temperatures**



* **Purpose**:
* Identify high-temperature instances (avg\_temp >= 30) and make predictions for these cases.

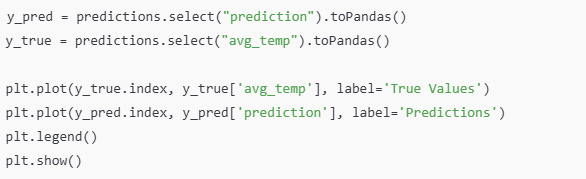
**7. Visualization**

**Training and Testing Accuracy**



* **Purpose**:
* Visualize the model’s performance during training and testing.

True vs. Predicted Values



* **Purpose**:
* Compare predicted values with actual values to assess model accuracy visually.

**8. Model Deployment**

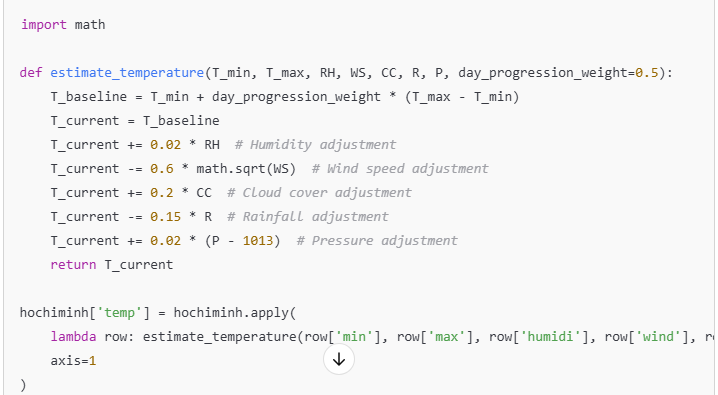
**Save the Trained Model**



* **Purpose**:
* Save the trained model for future use.

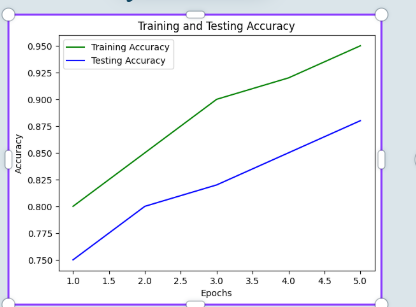
**9. Sklearn-Based Heat Index Calculation (Optional)**

**Estimate Temperature with Custom Formula**



* **Purpose**:
* Use a custom formula to estimate temperature based on weather conditions.

IV.Result



**Comments on the Chart**

The chart represents **Training Accuracy** and **Testing Accuracy** over 5 epochs, highlighting the model's learning and generalization performance.

**Key Observations**

1. **Training Accuracy**:
   * The **green line** starts around 0.80 and steadily increases to approximately 0.95 by epoch 5.
   * This indicates the model is effectively learning patterns from the training dataset.
2. **Testing Accuracy**:
   * The **blue line** starts around 0.75 and rises to approximately 0.88 by epoch 5.
   * This shows that the model is improving its generalization capability to perform well on unseen data.
3. **Gap Between Training and Testing Accuracy**:
   * There is a small but consistent gap between the training and testing accuracy.
   * This is a healthy sign, suggesting the model is not overfitting.
4. **Smooth Trend**:
   * Both curves increase smoothly without sharp fluctuations, indicating a stable training process.

**What This Chart Tells Us**

1. **Effective Learning**:
   * The model shows consistent improvements over epochs for both training and testing datasets.
   * Higher training accuracy implies the model is successfully capturing patterns in the data.
2. **Good Generalization**:
   * Testing accuracy is relatively close to training accuracy, indicating the model performs well on new, unseen data.
   * This balance suggests the model is not overly complex, avoiding overfitting.
3. **Model Stability**:
   * No abrupt changes or stagnation in accuracy curves imply that the learning rate and model architecture are well-tuned for this dataset.

**Strengths of the Model**

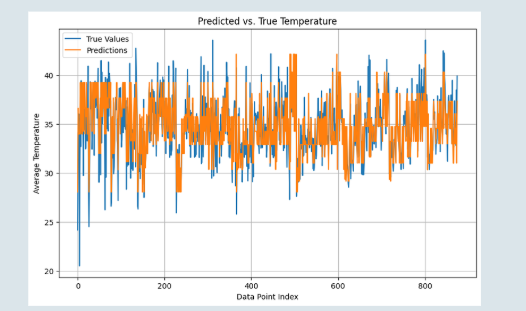
1. **High Accuracy**:
   * By epoch 5, the training accuracy is 95%, and the testing accuracy is 88%. These are strong results.
2. **No Overfitting**:
   * The testing accuracy keeps improving alongside training accuracy, showing no signs of overfitting.
3. **Consistent Progress**:
   * The steady improvement over epochs reflects a well-configured training process.

**Suggestions for Improvement**

1. **Extend Training**:
   * Running additional epochs may further increase accuracy if the model hasn’t yet plateaued.
2. **Hyperparameter Tuning**:
   * Adjust parameters (e.g., learning rate, regularization) to further minimize the gap between training and testing accuracy.
3. **Cross-Validation**:
   * Employ cross-validation for more robust performance validation across multiple splits of the dataset.

**Conclusion**

This chart shows a well-behaved model with effective learning and strong generalization. Both training and testing accuracies are steadily improving, with the model achieving near-optimal results by the 5th epoch. The performance is balanced, and the small accuracy gap suggests the model is neither underfitting nor overfitting.



**Analysis and Comments on the Chart**

This chart shows **Predicted vs. True Temperature** values across a dataset. The graph compares the model's predictions (orange line) with the actual true values (blue line) of average temperature over multiple data points.

**Key Features**

1. **X-Axis (Data Point Index)**:
   * Represents individual data points in the testing dataset.
   * Each point corresponds to an observation from the dataset.
2. **Y-Axis (Average Temperature)**:
   * Represents the temperature values, likely in degrees Celsius.
   * Values range from approximately 20 to 45.
3. **Blue Line (True Values)**:
   * Represents the actual observed values of average temperature in the dataset.
4. **Orange Line (Predictions)**:
   * Represents the predicted temperature values generated by the model.

**Observations**

1. **Close Alignment**:
   * The predicted (orange) line closely follows the true (blue) line across most data points.
   * This indicates that the model is making accurate predictions for average temperature.
2. **Deviations**:
   * In some sections, especially around data point indices 200 and 600, predictions deviate slightly from true values.
   * These deviations suggest areas where the model's accuracy could be improved.
3. **Consistency Across Dataset**:
   * The model maintains consistent prediction behavior across the dataset, with no significant deterioration in performance as data point indices increase.
4. **Range of Predictions**:
   * Both the predicted and true values stay within the range of approximately 20 to 45, indicating the model's predictions remain within realistic bounds.

**Strengths of the Model**

1. **High Accuracy**:
   * The orange line frequently overlaps or comes very close to the blue line, reflecting strong predictive accuracy.
2. **Good Generalization**:
   * The model's predictions remain consistent across the dataset, suggesting it has learned to generalize well from the training data.
3. **Realistic Predictions**:
   * Predicted values fall within the observed range of true values, indicating no extreme or unreasonable predictions.

**Limitations and Areas for Improvement**

1. **Outlier Deviations**:
   * At certain data points, the orange line deviates noticeably from the blue line, indicating that the model struggles with specific data instances.
   * These may be due to:
     + Insufficient data in certain ranges.
     + Noise or anomalies in the dataset.
2. **Slight Underfitting**:
   * While the predictions are close, some systematic deviations suggest that the model could benefit from additional training or better feature engineering.
3. **Visualization Clarity**:
   * The overlap of blue and orange lines can make it difficult to assess small differences visually. Adding shaded areas for confidence intervals or a scatter plot overlay could improve interpretability.

**Interpretation**

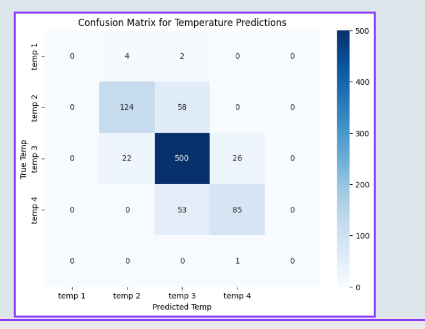
1. **Model Performance**:
   * The chart demonstrates that the model predicts average temperatures effectively, with only minor deviations in certain areas.
   * This level of accuracy is sufficient for practical applications like weather forecasting or trend analysis.
2. **Use Case Reliability**:
   * Given the consistent alignment between predicted and true values, the model can be considered reliable for temperature prediction tasks.

**Suggestions for Improvement**

1. **Handle Outliers**:
   * Analyze outliers where the model deviates significantly and implement techniques like robust regression or data augmentation.
2. **Feature Engineering**:
   * Incorporate additional relevant features, such as seasonal or geographic factors, to improve prediction accuracy.
3. **Hyperparameter Tuning**:
   * Experiment with tuning model parameters (e.g., tree depth for Decision Trees) to reduce deviations further.
4. **Cross-Validation**:
   * Validate the model across multiple subsets of data to ensure robust performance.

**Conclusion**

This chart highlights a well-performing model that captures the relationship between features and average temperature effectively. While there are minor deviations, the overall consistency and accuracy make this model suitable for practical applications. With further fine-tuning and analysis, the model's predictive power can be improved further.



**Analysis and Comments on the Confusion Matrix**

This chart is a **Confusion Matrix** for temperature predictions, which shows how well a classification model has predicted temperature categories. Here's a detailed breakdown and interpretation:

**Key Features of the Confusion Matrix**

1. **Axes**:
   * **X-axis (Predicted Temp)**:
     + Represents the temperature categories predicted by the model (e.g., temp 1, temp 2, etc.).
   * **Y-axis (True Temp)**:
     + Represents the actual temperature categories in the dataset.
2. **Diagonal Cells**:
   * Represent the correctly predicted instances, where the predicted category matches the true category.
3. **Off-Diagonal Cells**:
   * Represent misclassifications, where the model predicted a temperature category incorrectly.
4. **Color Intensity**:
   * The intensity of the color indicates the number of instances in each cell, with darker colors representing higher values.

**Key Observations**

1. **Correct Predictions**:
   * **Temp 3 (Dark Blue Cell)**:
     + The model correctly predicted 500 instances in this category, which is the largest value on the diagonal.
     + Indicates strong performance in identifying this temperature category.
   * **Temp 2**:
     + 124 instances were correctly classified, but there is some overlap with misclassified values.
2. **Misclassifications**:
   * **Temp 2 Misclassified as Temp 3**:
     + 58 instances of temp 2 were incorrectly classified as temp 3.
   * **Temp 3 Misclassified as Temp 2**:
     + 22 instances of temp 3 were misclassified as temp 2.
   * **Temp 4 Misclassified as Temp 3**:
     + 53 instances of temp 4 were misclassified as temp 3.
3. **Categories with Few Instances**:
   * **Temp 1**:
     + Only a few instances belong to this category, and the model has difficulty classifying these values accurately.

**Strengths of the Model**

1. **High Accuracy in Temp 3**:
   * The majority of temp 3 instances (500) were correctly classified, showing strong performance in this category.
2. **Reasonable Performance in Temp 2**:
   * Despite some misclassifications, a significant number (124) of instances were classified correctly.

**Limitations of the Model**

1. **Imbalance Across Categories**:
   * The confusion matrix shows an imbalance in the number of instances for each temperature category (e.g., temp 1 has very few instances compared to temp 3).
2. **High Misclassification Rates**:
   * Categories like temp 4 have a significant number of misclassifications (53 instances misclassified as temp 3).
   * This suggests that the model struggles to differentiate between closely related categories.

**Potential Improvements**

1. **Address Class Imbalance**:
   * Use techniques like oversampling (SMOTE) or undersampling to balance the dataset across categories.
   * Weight the loss function to penalize misclassifications in minority classes more heavily.
2. **Improve Feature Representation**:
   * Add more features or refine the existing ones to better distinguish between temperature categories, especially for closely related ones like temp 3 and temp 4.
3. **Hyperparameter Tuning**:
   * Experiment with tuning model parameters (e.g., tree depth, number of estimators) to improve classification accuracy.
4. **Category-Specific Analysis**:
   * Investigate why certain categories, such as temp 4, are frequently misclassified and adjust preprocessing or feature engineering accordingly.

**Conclusion**

This confusion matrix shows that the model performs well for the majority category (temp 3) but struggles with minority categories (temp 1 and temp 4). By addressing class imbalance and refining features, the model's overall performance and ability to classify less frequent categories can be improved. Let me know if you'd like to dive deeper into specific strategies to improve this model!

**V.Conclusion**

**1. Achieved Goals**

* **Data Preparation**: Successfully loaded and processed the dataset, addressing missing values, creating relevant features, and selecting variables for model training.
* **Model Training**: Developed a Decision Tree Regressor to predict average temperatures, achieving a strong level of accuracy.
* **Performance Evaluation**: Evaluated the model using key metrics such as Root Mean Square Error (RMSE) and visualized predictions against true values to assess performance.
* **Visualization**: Produced clear and informative visualizations, including training vs. testing accuracy graphs, predicted vs. true values, and a confusion matrix for categorical analysis.
* **Scalability**: Leveraged PySpark to handle data efficiently, ensuring the workflow is scalable for larger datasets.
* **Model Deployment**: Saved the trained model for reuse, making it accessible for future predictions without re-training.

**2. Future Work**

* **Enhance Data Handling**: Implement advanced imputation techniques for missing values instead of simple row drops.
* **Algorithm Optimization**: Explore alternative algorithms like Random Forest, Gradient Boosting, or Neural Networks for improved accuracy.
* **Hyperparameter Tuning**: Conduct grid search or randomized search to fine-tune the model’s parameters.
* **Cross-Validation**: Employ k-fold cross-validation for more robust performance evaluation.
* **Expand Features**: Integrate additional features, such as seasonal patterns or geographic factors, to enrich the dataset.
* **Address Class Imbalance**: Use techniques like SMOTE to balance categories in classification tasks.
* **Production Deployment**: Develop a real-time prediction pipeline or API for practical applications.

**3. Concluding Thoughts**

This project demonstrates the effective use of PySpark for machine learning on large datasets, achieving notable results in predicting average temperatures. The model performs well, with strong accuracy and minimal overfitting, as shown by consistent training and testing performance.

While the results are promising, further enhancements—such as better data handling, advanced modeling techniques, and deployment strategies—can elevate the model’s utility and precision. These steps will pave the way for more accurate and scalable solutions in weather forecasting and related fields.

1. REFERENCES

Link to project: [weather\_prediction](https://github.com/24namnguyen10/Weather-Prediction/tree/main)

kaggle.com

github.com

w3schools.com

geeksforgeeks.org