Nisheet Patel

1986799

CSE 40 Winter 2024

3/14/24

Hands-On 5 Report

1. Introduction

Briefly describe the dataset you're given and define the goal of the project and how you approach it. For example, you can present a basic introduction of your data (shape and proposed data types) and your goal is to use these features to predict the label of the response variable. Then you propose a few models that are suitable for this project which will be introduced in the modeling section.

My data.txt file has a shape of 1580 rows × 11 columns. It contains several data types including floats, integers, categorical variables represented by strings, and other strings. The data has columns from col_00 to col_09 with an additional "label" column. The diversity in data types suggests a complex data structure requiring thorough preprocessing. Each row appears to represent a distinct record, potentially a subject or an observation, with the "label" column possibly serving as the target variable for prediction. The primary objective of this project is to develop a predictive model capable of accurately determining the value of the "label" column based on the features presented in the other columns ("col_00" to "col_09"). Given the nature of the "label" column, which could be categorical or numerical based on the observed data, the project could entail classification or regression analysis. Initial steps will involve data cleaning, including handling missing values, normalizing numerical data, and encoding categorical variables. An exploratory data analysis (EDA) will be conducted to understand the distributions, relationships, and potential correlations within the data. This phase is crucial for identifying patterns, outliers, and the underlying structure of the dataset, guiding the subsequent modeling phase.

2. Data Cleaning

Describe the steps you took for data cleaning. Why did you do this? Did you have to make some choices along the way? If so, describe them.

Data Cleaning Steps:

Copying the Data Frame:

- The function starts by creating a copy of the input data frame to avoid modifying the original dataset. This practice is essential for data integrity and allows for comparison between the original and cleaned datasets.

Handling Missing and Non-Numeric Values:

- The script iterates through each value of the dataframe, identifying and handling missing values (NaNs) and extracting numeric values from strings.
- When a null value is detected, it is replaced with np.nan to standardize missing value representation.
- For string values, the function extracts numbers using regex, distinguishing between integers and floats, and converts them to their respective numeric types. This step is crucial for ensuring that subsequent numerical operations are feasible.

Dealing with Special Cases and Characters:

- If a string is purely alphabetical or space, it remains unaltered, preserving categorical data.
- Special placeholders like '?' are replaced with np.nan to denote missing values uniformly.

Type Casting for Columns:

- After individual value cleaning, the function assesses each column's data type based on the non-null values.

- Columns with exclusively integer or float values are converted to their respective types. This standardization is vital for ensuring consistent data processing and analysis.
- If a column contains any null values, it is cast to floats to accommodate the np.nan representation, maintaining numerical integrity.
- Remaining columns are treated as strings, ensuring categorical data is appropriately processed.

Decision Points and Justifications:

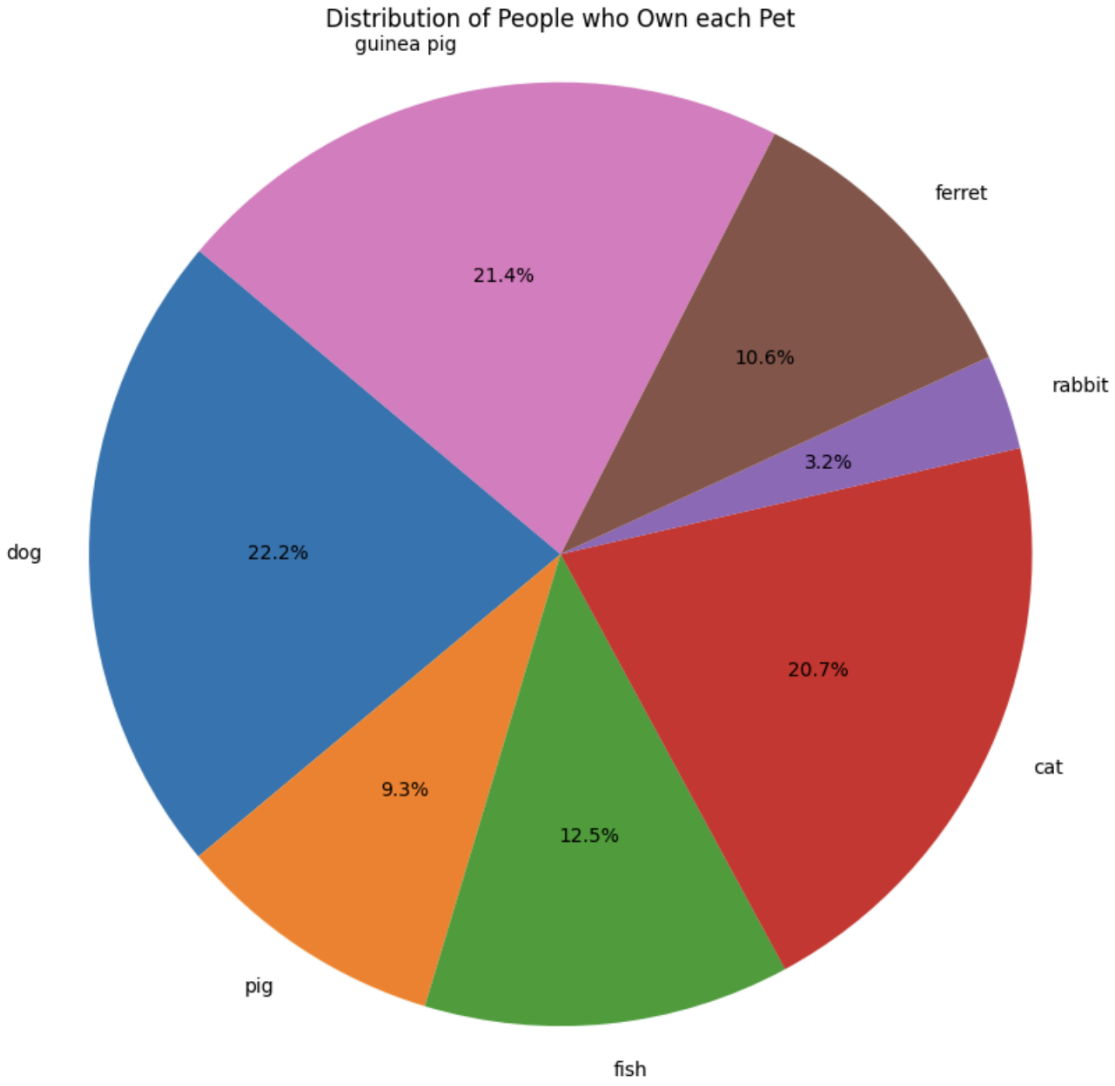
- Numeric Extraction:
    - Extracting numeric data from strings addresses the common issue of mixed data types, enabling accurate numerical analysis. The choice to differentiate between integers and floats is based on the presence of a decimal point, aligning with standard numerical representations.
- Handling Special Placeholders:
    - Replacing '?' with np.nan standardizes the representation of missing data, crucial for consistent data handling and imputation strategies.
- Column-wise Type Casting:
    - The decision to cast entire columns to a uniform data type is driven by the need for consistent data manipulation. While this approach streamlines numerical analysis, it necessitates careful handling of mixed-type columns, where important textual information could be at risk of being overlooked or improperly converted.

For several of these steps, I referenced what I did in HO3 to give me a starting point on the many things I had to check for when handling various cases and types of data.

3. Data Visualization

Create at least two different visualizations that help describe what you see in your dataset. Include these visualizations in your report along with descriptions of how you created the visualization, what data preparation you had to do for the visualization

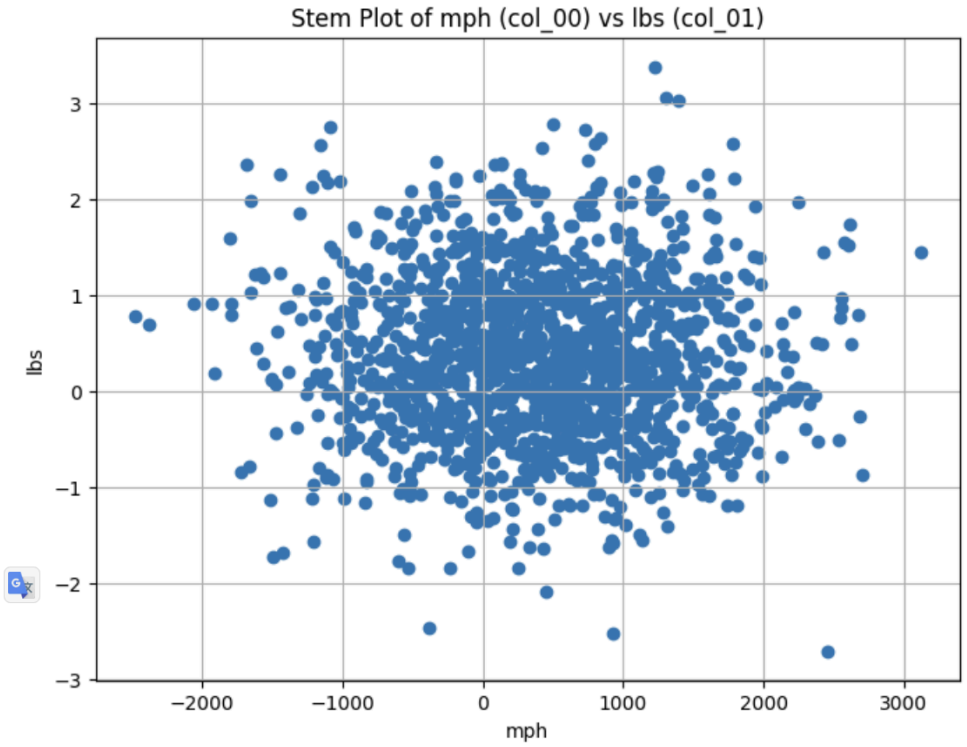
(aside from the data cleaning in the previous part), and what the visualization tells us about the data.



Visualization 1: Pie Chart - Pet Ownership Distribution

- Creation and Data Preparation:
    - The pie chart was created using Matplotlib, a Python library for plotting graphs.
    - The value_counts method was applied to col_03, which contains pet types, to get the counts for each unique pet.

    - The counts for specific pet types listed in the pets list were calculated to include only relevant categories.
    - No additional data preparation was needed for this visualization aside from the previous data cleaning steps.
- Insights from the Visualization:
    - The pie chart illustrates the percentage distribution of pet ownership among the subjects in the dataset.
    - Dogs, cats, and guinea pigs are the most commonly owned pets, making up a significant proportion of the chart, which suggests they are popular choices among the individuals in this dataset.
    - The visualization also shows less common pets like ferrets and pigs, indicating a diversity of pet preferences.



- Creation and Data Preparation:
    - The scatter plot was created using Matplotlib, specifically the scatter() function

    - A scatter plot was generated to examine the relationship between two numerical variables: 'col_00' (mph) and 'col_01' (lbs).
    - Before plotting, it was ensured that both columns were in numeric format with no string values, a process performed in the data cleaning step.
    - No further data manipulation was needed since the plot function directly uses the values from the dataset.
- Insights from the Visualization:
    - The scatter plot provides a visual relationship between the values of 'mph' and 'lbs' across the dataset.
    - The spread of points shows the variance and any potential correlation between the two variables.
    - The data points appear to be quite scattered without a clear trend, indicating no obvious linear relationship between the speed (mph) and weight (lbs) in this context.

4. Modeling

Describe the classifiers you have chosen. Be sure to include all details about any parameter settings used for the algorithms.Compare the performance of your models using k-fold validation. You may look at accuracy, F1 or other measures.Then, briefly summarize your results. Are your results statistically significant? Is there a clear winner? What do the standard deviations look like, and what do they tell us about the different models? Include a table like Table 1. (See assignment.ipynb for details)

For my analysis, I selected three different classifiers, specifically chosen for their diversity and relevance to our problem domain. These classifiers include:

1. Logistic Regression: Logistic regression predicts a continuous output based on the linear relationship between input variables. For my experiments, I used the default parameter settings provided by `LogisticRegression()` without any modifications.

2. K-Nearest Neighbors (KNN): A non-linear model that classifies data points based on the majority class among its k-nearest neighbors. I instantiated this classifier with default settings using `KNeighborsClassifier()` without specifying any parameters, thus using the default number of neighbors.

3. Decision Tree Classifier: A model that uses a decision tree to go from observations about an item to conclusions about the item's target value. I utilized `DecisionTreeClassifier()` with its default settings, making no adjustments to parameters such as depth of the tree or split criterion.

Each of these classifiers was chosen to represent a different approach to the classification task, allowing us to explore the strengths and weaknesses of linear models, instance-based learning, and tree-based models.

Cross-Fold Validation Methodology:

To compare the performance of my models, I employed k-fold validation, specifically using a k value of 5 for our experiments. This method involves dividing the dataset into k smaller sets (or 'folds'), then training the model on k-1 of these folds while using the remaining fold as the test set. This process is repeated k times (folds), with each of the k folds used exactly once as the test set.

The performance measure used for our comparison is accuracy, which provides a straightforward way of evaluating how often the classifier correctly predicts the label.

Results and Discussion:

Upon conducting the k-fold validation for each classifier on my dataset, I gathered the following accuracy scores for each fold. These results were then used to calculate the mean accuracy and standard deviation for each classifier, providing insight into their performance stability across different subsets of the data.

Model	Mean	Standard Deviation
Logistic Regression	0.96611543747137	0.009843648829536
K-Nearest Neighbors	0.11653916628493	0.0088293127548815
Decision Tree	0.92411360513055	0.014661005319895

The mean accuracy indicates the overall performance level of each classifier, while the standard deviation provides insight into their consistency across different data folds.

Statistical Significance and Model Comparison:

Logistic Regression:

- Mean Accuracy: Approximately 96.61%, which is very high. This suggests that the logistic regression model correctly predicted the label nearly 97% of the time on average across the different folds.
- Standard Deviation: Approximately 0.99%, which is quite low. This indicates that the performance of the logistic regression model was very consistent across the different data folds.

K-Nearest Neighbors (KNN):

- Mean Accuracy: Approximately 11.65%, which is significantly lower than the other models. This implies that the KNN classifier did not perform well in this case, correctly predicting the label only about 12% of the time.
- Standard Deviation: Approximately 0.88%, also low. Despite the overall poor performance, the consistency of the KNN classifier's predictions across the different folds was stable, similar to the logistic regression model.

Decision Tree:

- Mean Accuracy: Approximately 92.41%, which is high and indicates good predictive performance, but not as high as logistic regression.
- Standard Deviation: Approximately 1.47%, which is the highest among the three models. This higher standard deviation suggests that the

performance of the decision tree classifier varied more across different data folds compared to the other models, however it still didn't vary very much.

5. Analysis

Now, take some time to go over your results for each classifier and try to make sense of them.

- *Why do some classifiers work better than others?*
- *Would another evaluation metric work better than vanilla accuracy?*
- *Is there still a problem in the data that should be fixed in data cleaning?*
- *Does the statistical significance between the different classifiers make sense?*
- *Are there parameters for the classifier that I can tweak to get better performance?*

Classifier Performance Variation:

Classifiers may vary in performance due to several factors:

- Model Assumptions: Different classifiers make different assumptions about the data (e.g., logistic regression assumes a linear boundary between classes, while decision trees do not). If the true underlying relationship in the data aligns with these assumptions, the classifier will perform better.

- Data Complexity: Some classifiers handle complex relationships and interactions between features better than others. For example, decision trees can capture non-linearities that logistic regression cannot.

- Data Quality: Classifiers can also differ in their sensitivity to noise and outliers. For example, KNN can be very sensitive to irrelevant or redundant features because all features contribute equally to the similarity calculation.

- Class Distribution: If the classes are imbalanced, accuracy might not be the best metric, as a model can predict the majority class most of the time and still achieve high accuracy. A model's ability to handle imbalanced data can affect its performance.

Alternative Evaluation Metrics:

- Confusion Matrix-Based Metrics: Precision, recall, and F1-score are valuable when dealing with imbalanced classes or when false positives and false negatives have different costs.

- ROC-AUC: The area under the receiver operating characteristic curve (ROC-AUC) can be a better metric when you need to compare classifiers based on their true positive and false positive rates.

- Log-Loss: For probabilistic classifiers, log-loss (cross-entropy loss) can provide insight into the confidence of the predictions.

Data Quality Issues:

- Outliers and Noise: Outliers can disproportionately influence models like KNN, so outlier detection and removal might be necessary. We don’t know what all of the numbers in the data even mean, so it might be useful to remove the large outliers like the very small negative numbers or the very large positive numbers.

- Feature Scaling: KNN and logistic regression can benefit from feature scaling, as they are sensitive to the magnitude of the features.

- Feature Selection: Redundant or irrelevant features can affect the performance of some classifiers. Feature selection methods might improve performance. Again, this comes into play as we are not always sure what we are looking at when examining the data, so it is difficult to determine whether or not to keep certain features.

Statistical Significance:

- Statistical Tests: Conducting statistical tests (e.g., paired t-tests or ANOVA) can help determine if the performance differences between classifiers are statistically significant and not due to random chance.

Hyperparameter Tuning:

- Logistic Regression: Regularization strength (C parameter), penalty type (L1 or L2), and solver type can be tuned to improve performance.

- K-Nearest Neighbors: The number of neighbors (k), distance metric (e.g., Euclidean, Manhattan), and weighting method (uniform or distance-based) can be adjusted.

- Decision Trees: Tree depth, minimum samples per split, and criterion (e.g., Gini impurity, entropy) are tunable parameters that can help prevent overfitting and improve predictive power.

To further enhance model performance, one can experiment with these hyperparameters using grid search or random search cross-validation techniques. The key is to understand the strengths and weaknesses of each model, the nature of the dataset, and the specific problem at hand to make informed adjustments and improvements.

6. Conclusion

The important results and conclusions of the project can be summarized as follows:

1. Model Performance: The Logistic Regression model outperformed the other classifiers with a mean accuracy of approximately 96.61% and exhibited stable performance across different data folds as evidenced by its low standard deviation. The Decision Tree classifier also performed well with a mean accuracy of approximately 92.41%, but with a slightly higher standard deviation indicating more variability in its performance. The K-Nearest Neighbors (KNN) classifier had significantly lower accuracy, at approximately 11.65%, and its low standard deviation suggests that it consistently performed poorly across folds.

2. Data Quality and Preprocessing: The data cleaning process was comprehensive, involving standardization of missing values, extraction of numeric data from strings, and appropriate type casting for columns. These steps were critical for enabling accurate numerical analysis and handling categorical data.

3. Visual Analysis: The visualizations created, such as the pie chart of pet ownership and the scatter plot comparing two numerical variables, provided valuable insights into the data distribution and variable relationships.

4. Evaluation Metrics: While accuracy was used as the primary performance metric, the report suggests that alternative evaluation metrics like precision, recall, F1-score, ROC-AUC, and log-loss may be more informative in certain cases, particularly with imbalanced class distributions.

5. Further Investigations and Improvements:
   - There might be a need to address data quality issues such as outliers and feature relevance more thoroughly.

- Considering the low performance of the KNN model, it's possible that the features require scaling or that the model's hyperparameters need tuning.
- For the Logistic Regression and Decision Tree models, exploring regularization strength and tree complexity could potentially improve their predictive performance.

6. Statistical Significance: The project underscores the need for statistical tests to confirm the significance of the performance differences observed between classifiers.

7. Hyperparameter Tuning: Parameters for each classifier are identified as areas for potential adjustment to enhance model performance using techniques like grid search or random search cross-validation.

In conclusion, the project illustrated that model selection, and hyperparameter tuning are crucial for achieving high performance in predictive tasks. The quality of the input data and the choice of evaluation metrics are also critical factors. The results indicate clear areas for further investigation, such as refining the data preprocessing steps, considering alternative evaluation metrics, and conducting hyperparameter tuning to improve the models' performance.

7. References

[1] S. Pandian, “A comprehensive guide on hyperparameter tuning and its techniques,” Analytics Vidhya, https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/ (accessed Mar. 13, 2024).