

REPORT OF ASSIGNMENT 6

Carnegie Mellon University Africa

Submitted By: Peace Ekundayo Bakare

Course: Data, Inference, and Applied Machine Learning

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

Libraries Used:

1. pandas as peacepd
2. seaborn as peacesns
3. numpy as peacenp
4. matplotlib.pyplot as peaceplt
5. statsmodel.api as peacestats
6. mean_squared_error, confusion_matrix, accuracy_score from sklearn.metrics
7. SequentialFeatureSelector from sklearn.feature_selection
8. LogisticRegression, LinearRegression from sklearn.linear_model
9. train_test_split from sklearn.model_selection
10. LabelEncoder from sklearn.preprocessing

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

QUESTION 1 – NONLINEARITY

1.1 Explanation of Nonlinearity and reason to consider nonlinear relationships between variables

Mathematically, Nonlinearity is a term used to describe the relationship between a dependent variable and an independent variable which is not predictable from a straight line. Statistically, Nonlinearity is a term used to describe a situation when there is no direct relationship between an independent variable and a dependent variable. i.e. changes in the input does not infer direct proportion of change in the output variable.[1]

A nonlinear function can take many forms, such as rational, polynomial, exponential, and logarithmic functions. The slope of a nonlinear function varies at different points along its curve. Calculus and other advanced mathematical techniques can be used to model and analyze nonlinear functions.[2]

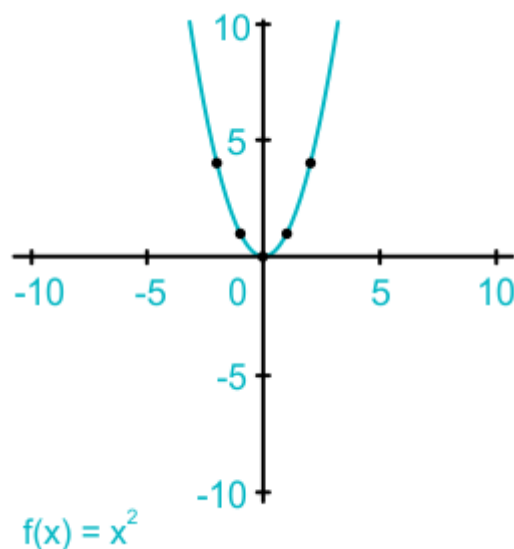
Examples of a nonlinear equation are:

$$f(x) = x^2, f(x) = 3^x - 7, f(x) = \sin x$$

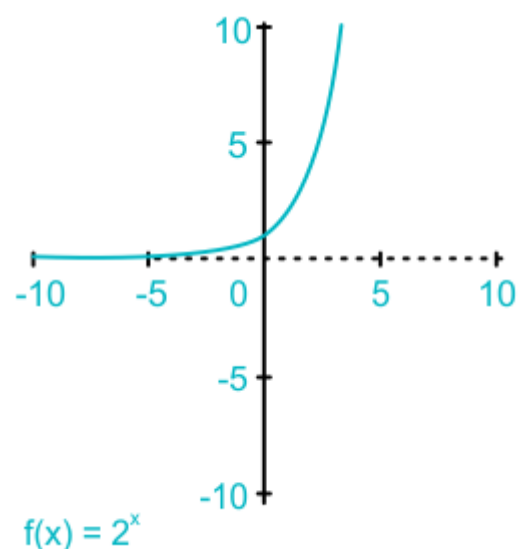
The graph of a nonlinear function is not a straight line. It can be a curve

Graphs that depict nonlinear functions are:

Parabolic Graph



Exponential Graph



The graphs are taken from [2]

Why might it be necessary to consider nonlinear relationships between variables?

Nonlinear relationships between variables is essential to be considered to accurately model the relationship. It allows for a more nuanced understanding of complex interactions and leads to better predictions and risk assessments.

Here is a few importance of considering Nonlinear relationships between variables:

1. **Realistic Modeling:** Many processes in nature and culture are naturally nonlinear. For example, financial products such as options are nonlinear derivatives in which changes in the underlying variables do not result in proportional changes in value. Ignoring nonlinearity can result in oversimplified models that fail to capture key dynamics.
2. **Enhanced Predictive Power:** By including nonlinear linkages into models, analysts can better forecast outcomes and comprehend complicated interactions. This is especially important in sectors like finance, where risk assessment models need to account for the unpredictable character of asset returns.
3. **Accurate Risk Management:** Recognizing nonlinearity is critical for successful risk management in finance. Investors frequently employ complex approaches (e.g., Monte Carlo simulations) to assess prospective losses or gains associated with nonlinear assets, ensuring that they are prepared for diverse market scenarios.
4. **Complex Systems Understanding:** Nonlinear analysis helps academics and practitioners comprehend complex systems across fields, such as biology, economics, or physics, when basic linear assumptions might lead to inaccurate findings or inefficient solutions.

1.2 Mathematical Equation for a nonlinear model and examples of an application where it might be appropriate

Nonlinear model equations can be generally represented as:

$$y = f(x, \beta) + \varepsilon$$

Where f is a nonlinear function, x are the independent variables,

β are the parameters, and ε is the error term

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

An example of an application where nonlinear model might be appropriate is the modeling population growth using the logistic function:

$$P(t) = \frac{K}{(1 + Ae^{(-rt)})}$$

Where $P(t)$ is the population at time t , K is the carrying capacity,
 A and r are constants.

$$A \text{ is calculated as } \frac{K - P_0}{P_0}$$

1.3 Can a nonlinear model be more parsimonious than a linear model? Support your answer with a mathematical formula for both linear and nonlinear models

YES! A nonlinear model can be more parsimonious than a linear model because they can capture complex relationships with fewer parameters.

Mathematical formulae for linear and nonlinear models:

Linear Model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Nonlinear Model:

$$y = \beta_0 + \beta_1 e^{-\beta_2 x_1} + \varepsilon$$

The nonlinear model uses fewer parameters to potentially capture more complex relationships.

1.4 What characteristics are typically preserved when generating surrogates? Give the names of two surrogate techniques and describe the approaches for implementing them.

The characteristics that are typically preserved when generating surrogates are:

- Amplitude distribution
- Power Spectrum
- Autocorrelation structure
- Fourier Transform
- Mean
- Variance

The two surrogate techniques are:

1. Phase Randomization

Approach

This technique involves transforming the original time series into the frequency domain using a Fourier Transform. The magnitudes of the Fourier coefficients are retained, while their phases are randomized uniformly. After this transformation, an inverse Fourier Transform is applied to generate the surrogate data. This method preserves the linear correlation structure of the original series while destroying any nonlinear characteristics.

Implementation

- Computation of the Fourier Transform of the original time series
- Retain the magnitudes of the Fourier coefficients
- Randomize the phases of these coefficients
- Apply the inverse Fourier Transform to obtain the surrogate data

2. Iterative Amplitude Adjusted Fourier Transform (IAAFT)

Approach

This technique aims to preserve both the linear structure and the amplitude distribution of the original time series. It involves two main steps:

- First, scaling the data to approximate a Gaussian distribution (Gaussianization),
- Second, applying a modified Fourier Transform process like the RS but adjusting for amplitude.

Implementation

- Transform the original data to a Gaussian distribution through a suitable transformation
- Perform a Fourier Transform on this Gaussianized data
- Randomize the phases of the Fourier coefficients while keeping their magnitudes intact
- Inverse transform back to obtain surrogate data, adjusting amplitudes to match those of the original series iteratively

1.5 Information, Entropy and Mutual Information

Information

Information measures the reduction in uncertainty about an outcome when a particular event occurs.

It can be represented mathematically as:

$$I(p) = \log(1/p) = -\log(p)$$

where p is the probability of observing an event x .

Entropy

Consider \mathbf{X} to be a discrete random variable, and that we are in a “perfect world”. The entropy can be thought of as the amount of randomness in \mathbf{X} (in bits). It quantifies the amount of information needed to describe the state of the system. In the context of probability distributions, entropy indicates how spread out the distribution is.

Mathematical definition of entropy is:

$$H(x) = - \int p_x(x) \log[p_x(x)] dx$$

$$H(X) = - \sum_i p(x_i) \log(p(x_i))$$

Where $p(x_i)$ is the probability of an event x_i .

Mutual Information

Mutual Information measures the amount of information that one random variable contains about another. It quantifies the reduction in uncertainty of one variable due to the knowledge of the other. In other words, mutual information captures the dependence between two variables.

The mutual information $I(X; Y)$ between X and Y is defined as:

$$I(x, y) = H(x) + H(y) - H(x, y)$$

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x) p(y)} \right)$$

where $p(x, y)$ is the joint probability distribution of XX and YY . $p(x)$ and $p(y)$ are the marginal probability distributions of X and Y , respectively.

Entropy to Measure Regularity

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

Entropy can be used as feature for measuring the regularity of a time series or other data sequence. For example, approximate entropy or sample entropy quantifies the complexity or regularity of a signal by measuring the likelihood of similar patterns over time. Lower entropy suggests that there is higher regularity or a more predictable system, while higher entropy suggests a chaotic system.

Example: In physiological monitoring, entropy measures are used to analyze heart rate variability. Low entropy in heart rate signals may indicate regular, predictable rhythms associated with good health, while higher entropy may signal irregularities or anomalies, potentially pointing to underlying health issues.

Mutual Information for Feature Selection

Mutual information is used in feature selection as it measures the dependency between two variables. We use mutual information to identify the most relevant features for predicting the target variable. Features with high mutual information scores are preferred because they reduce the uncertainty about the target variable.

The process involves:

- Compute mutual information between features and target – Calculate $I(X; Y)$
- Select Relevant Features – set a threshold for mutual information and include only features above the threshold, and then rank features by their MI scores to select top features
- Optionally, iterative selection is done by applying conditional MI to account for redundancy among selected features. The features that add unique information about the target are selected.

Why Mutual Information is Better than Correlation:

- **Nonlinear Relationships:** Mutual information can detect associations that are not linear, unlike correlation, which only measures the strength of linear relationships.
- **Complex Dependency:** Mutual information is sensitive to any form of dependency between variables, providing a more comprehensive measure of association
- **Multi-modal Distributions:** Mutual Information does not assume that data distributions are unimodal, and Gaussian like correlation does. It works effectively for multi-modal relationships

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

QUESTION 2 – CLASSIFICATION USING TREES

2.1 Decision Trees

A Decision Tree is a non-parametric supervised machine learning algorithm used for classification and regression tasks. It has a hierarchical tree structure that includes a root node, branches, internal nodes, and leaf nodes.[3]

Nodes

- **Root Node:** This is the topmost node representing the initial decision point on a feature of the dataset. It represents the best feature to split the data.
- **Internal Nodes:** These are decision points where the data is split based on specific feature values. Each internal node applies a test on a feature, dividing the dataset into branches based on the outcome of that test. Represent the features used for splitting the data based on specific decision rules.
- **Leaf Nodes:** These are the final nodes of the tree where the outcome (classification or decision) is assigned. Leaf nodes represent the end of a particular decision path in the tree. Terminal nodes that represent the predicted outcome (class label or numerical value).

Branches

Branches connect the nodes and represent the outcome of the tests applied at each internal node. Each branch leads to another node or a leaf, showing the possible paths, the decision-making process can take based on the feature values.[4]
Connections between nodes representing the possible values of the features.

Pruning of Decision Trees and why it is necessary?

Pruning of Decision Tree is a technique used in machine learning to optimize decision tree models to avoid or reduce overfitting while improving the ability of the model to generalize to new data. Pruning removes parts of the decision tree that do not provide significant predictive power.[5] There are two types of pruning: Pre-pruning (Early Stopping) and Post-pruning.

Why is it necessary to prune a tree?

- **Prevention of Overfitting:** Poor generalization is the result of overfitting of decision tree models on training dataset. Pruning helps to reduce the

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

complexity of the model and prevent it from fitting the noise in the training dataset.

- **Improving Generalization:** Pruned decision trees perform better on unseen data than complex decision trees. Pruning enhances the model's ability to perform well on data outside of the ones used for training.
- **Reduces Model Complexity:** The result of pruning is a simpler tree with fewer branches and nodes (internal, leaf), making it easier to interpret and understand.
- **Less Computational Resources:** A complex decision tree would likely consume much computational resources while a pruned tree would require less computational resources, which also helps to gain speed of training and making faster and accurate predictions.

Why are decision trees an attractive method for classification in practice?

Decision Trees are an attractive method for classification in practical applications because of the following reasons:

- **Feature Selection:** Decision Trees inherently performs feature selection, selecting the most significant features for making predictions.
- **Efficiency:** Decision Trees are relatively fast to train and make predictions, especially when they are pruned. They are also efficient for large datasets.
- **Robustness:** Decision Trees can handle noisy data, and they are less sensitive to outliers when compared to some other algorithms.
- **Handling Mixed Data Types:** Decision Trees can work with both categorical and numerical data types, making them very versatile for various types of features.
- **Ease of Interpretation:** Decision Trees structure provides a clear visual representation of the decision-making process, making them easy to understand by anyone.

Nonlinearity: Decision Trees are capable of capturing complex, nonlinear relationships between features and target variables.

2.2 Improving Existing Approach by constructing a Data-Driven Classifier

To improve the existing approach by constructing a data-driven classifier, the following steps are used:

1. **Understand the Current Rule-Based System:** Review the existing rules in the system to identify the limitations or gaps.
2. **Data Collection and Preparation:** Gather a robust dataset which includes input features and corresponding class labels. Perform Data cleaning to handle missing values (NaN), outliers, and inconsistent data and then split the dataset into training, validation, and test sets to ensure unbiased evaluation of the model.
3. **Feature Engineering:** Analyze the dataset to identify relevant features that could improve model performance. Also, create new features through transformations or combinations of existing ones if necessary.
4. **Model Selection:** Select either of decision trees, random forests, support vector machine or any other appropriate machine learning algorithms to use for classification. Then, start with simpler models then use complex ones if needed.
5. **Model Training:** Train the model that is selected per time using the separate training data. Utilize cross-validation techniques to optimize hyperparameters and prevent overfitting.
6. **Model Evaluation:** Evaluate the model performance on the validation set using metrics like accuracy, precision, F1-score, AUC-ROC, and recall. Models can also be compared to select the best performing ones.
7. **Model Tuning:** Fine-tune the model parameters based on validation results to enhance performance, and then use techniques such as regularization to ensemble methods if necessary.
8. **Testing and Validation:** Test the final model on an independent test set to assess its ability to generalize. Also, ensure that the model performs well on unseen data.

Testing the Validity of the New Model

- **Performance Evaluation on Test Dataset:** Ensure that the model generalizes well on unseen data by assessing its accuracy, F1-score, precision and other necessary metrics

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

- **Cross-Validation:** To ensure robustness and consistency across data splits, use k-fold cross-validation
- **Performance Comparison:** To verify that the performance improvement over the rule-based system is statistically significant, use statistical tests such as paired t-tests
- **Real-World Simulation and Benchmarking:** To measure practical performance, deploy the model in a controlled environment that simulates real-world settings. Test the model on edge cases and scenarios that the rule-based approach previously struggled with, to quantify improvements
- **Test Explainability and Monitor Over Time:** Obtain interpretable results from the data-driven model and conduct performance monitoring to validate the model's stability
- **Ethical and Regulatory Compliance:** Ensure that the model complies with relevant ethical and regulatory guidelines

2.3 Classifying the likelihood of survival using the Titanic Dataset

Explanation of the processes involved in steps

STEP 1: Loading the Titanic Dataset

I loaded the *titanic3.csv* file into the Google Colaboratory tool using *pandas.csv()* function. The dataset contained 1309 entries. It also has NaN values. I have used the *head()* and *info()* functions to understand the dataset.

STEP 2: Data Preprocessing

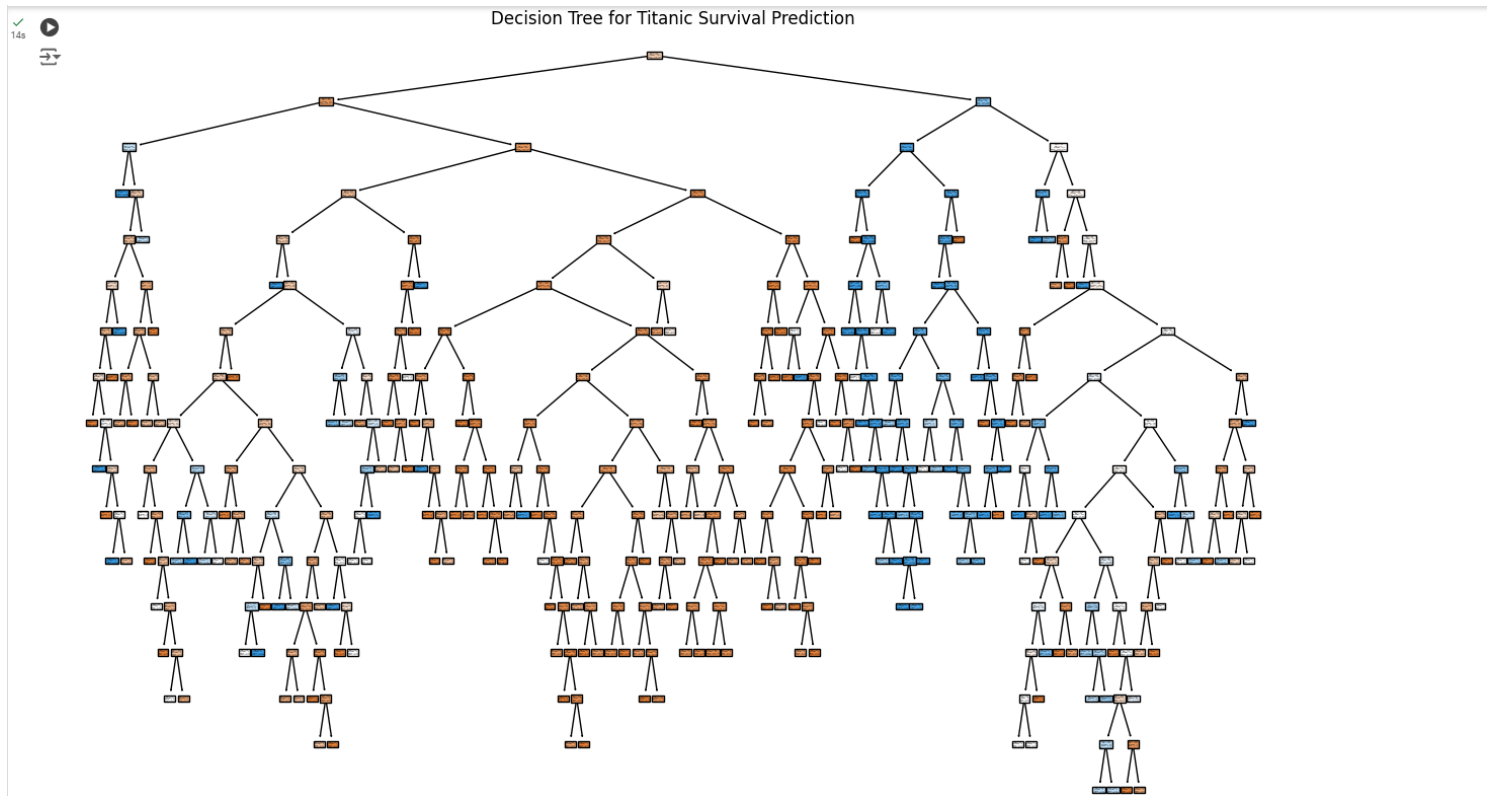
I selected 'pclass', 'sex' and 'age' columns as the predictor variables while I selected 'survived' as the target variable, which is the dependent variable to be predicted.

I handled the missing values in the 'age' column, replacing them with the mean. Also, I transformed the categorical feature, 'sex' column to numerical values, using 0 to represent male, and 1 to represent female. Lastly, I selected the X and y variables to be used in the next step.

STEP 3: Fit and Evaluate the Decision Tree

I used a 5-fold cross-validation to evaluate the decision tree's accuracy, train the decision tree classifier with a random state of 42 to increase its reproducibility. I then plotted a decision tree structure for visualization.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.



I visualized the tree using `plot_tree()` function.

2.4 Evaluate the performance of the Tree Before and After Pruning

I evaluated the performance of the tree before and after pruning using the in-sample and cross-validation techniques. I also provided Accuracy to help make some inferences.

BEFORE PRUNING

Accuracy before pruning is **84.6%** and the In-Sample Misclassification Error is **0.15**

```
➡ Accuracy before pruning: 0.8464  
   In-sample Misclassification Error: 0.1536
```

The Cross-Validation Misclassification Error is **0.2223**

```
➡ Cross-Validation Misclassification Error: 0.2223
```

To obtain the metrics of after pruning, I used cost complexity pruning to compute a range of `ccp_alpha` values. After which I trained multiple trees with different alpha values. In addition, I selected the alpha with the best cross-validation score, then

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

used it to train the pruned tree with a cost complexity pruning of the best alpha. The Best alpha score is **0.00106**.

```
Best Pruning alpha: 0.0010630758327427395
```

AFTER PRUNING

After pruning, the accuracy and the pruned decision tree misclassification error is captured below:

Accuracy after pruning is **81.21%**
Pruned Decision Tree Cross-Validation Error is **0.2193**
Pruned Decision Tree In-Sample Validation Error is **0.1879**

```
Pruned Decision Tree Cross-Validation Error: 0.2193
Accuracy after pruning: 0.8121
Pruned Decision Tree Misclassification Error: 0.1879
```

2.5 Compare the performance of the Final Tree with Logistic Regression

I trained the logistic regression model to compare its performance with the Pruned Decision Tree. I evaluated the performance of the Logistic Regression using Cross-Validation to measure generalization and got an **accuracy of 78.53%**. The In-Sample misclassification error is **0.2147** while the cross-validation misclassification error is **0.2239**.

Summary of the Data Values

Metric	Decision Tree	Pruned Decision Tree	Logistic Regression
Accuracy	0.8464476699770818	0.812070282658518	0.7853323147440795
In-Sample Misclassification Error	0.15355233002291824	0.187929717341482	0.2146676852559205
Cross-Validation Misclassification Error	0.22233921207335272	0.2192945511976837	0.22388347810827403

Comment on Result

The decision tree before pruning shows strong accuracy but exhibits signs of overfitting, as indicated by the lower in-sample error compared to the cross-validation error. Pruning reduces accuracy slightly but improves generalization by simplifying the tree and lowering the cross-validation misclassification error. Logistic regression, while achieving slightly lower accuracy than the decision tree, demonstrates consistent generalization

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

with closely aligned in-sample and cross-validation errors, though its inability to model complex nonlinear relationships limits its performance on this dataset.

Comment on Advantages and Disadvantages of Decision Trees and Logistic Regression

Advantages of Decision Tree (Pruned):

- **Nonlinear Relationship Handling:** Pruned Decision Trees captures interactions between features that logistic regression cannot. For example, pclass and age may interact in predicting survival.
- **Interpretability:** Decision trees are easier to interpret and visualize, which is beneficial for understanding feature importance.

Disadvantages of Decision Tree (Pruned):

- **Accuracy Trade-off:** Pruning reduced accuracy slightly compared to the unpruned tree.
- **Overfitting Risk (Unpruned):** Without pruning, decision trees can overfit the data, leading to poorer performance on new data.

Advantages of Logistic Regression:

- **Simplicity:** Easier to implement and computationally efficient.
- **Regularization-Friendly:** It can include L1/L2 regularization to prevent overfitting.
- **Linearity Assumption:** Works well when the relationship between features and target is linear.

Disadvantages of Logistic Regression:

- **Lack of Nonlinear Capability:** Cannot capture nonlinear interactions without feature engineering or polynomial terms.

BEST MODEL TO COMPETE IN THE KAGGLE COMPETITION IS THE PRUNED DECISION TREE.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

QUESTION 3 – CLASSIFICATION WITH KNN

3.1 Description of the concept behind focusing on small neighborhoods of state to construct parsimonious models & step-by-step procedure for implementing the model.

The concept behind constructing parsimonious models by focusing on small neighborhoods of state space is rooted in the idea of local modeling. This approach aims to create simpler models that focus on specific regions of the data, allowing for greater accuracy and interpretability without overfitting.[6]

There are key concepts to examine here:

- **State Space:** This refers to the set of all possible states or configurations that a system can take. In machine learning, this could be the feature space defined by the input variables.
- **Small Neighborhoods:** Instead of modeling the entire state space with a single complex model, the principle here is to break it down into smaller, more manageable regions (local neighborhoods).
- **Local Models:** In each neighborhood, we construct a simple model (e.g., linear or low-order polynomial), which is easier to interpret and avoids overfitting.

By focusing on small neighborhoods, we can build models that are both parsimonious (i.e., using fewer parameters) and accurate, as they capture the local structure of the data without unnecessary complexity.[6]

This method is particularly useful when the system exhibits complex, nonlinear behavior globally but can be approximated more simply in local regions. This concept is related to several techniques in machine learning and dynamical systems analysis, including:

- Local linear embedding
- Nearest neighbor methods
- Piecewise linear approximations
- Locally weighted regression

Step-by-Step Procedures [7]

Here are 10 steps to guide the implementation of the model

Step 1: Data Collection and Preprocessing

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

This step involves the **collection of Dataset** that represent the system's behavior across its state space, **cleaning of the dataset** i.e. handling missing values, outliers, and **feature scaling** which largely depends on the model type.

Step 2: State Space Partitioning

The State Space can be divided into smaller regions or neighborhoods. Some techniques that can be used are:

- **Grid-Based Partitioning** – Divide the feature space into uniform grids
- **Clustering Algorithms** – The use of clustering algorithms such as k-means
- **Adaptive Partitioning** – Partitioning based on data density or other criteria dynamically

Step 3: Local Model Selection

This step involves the selection of an appropriate model for each neighborhood. The choice of model depends on the nature of the data in the region. Linear model goes for a linear region, polynomial models are ideal for mild nonlinearity in environments and Decision Trees/Rule-Based Models are ideal for capturing decision boundaries.

Step 4: Model Fitting

This step involves fitting the selected local model using only the data points within each neighborhood region. For linear models, Least Squares regression is ideal, and for decision trees, a decision tree should be fitted into each local region.

Step 5: Neighborhood Weighting

Weights are assigned to data points based on their distance from the center of each neighborhood. Points closer to the center have higher influence on the local model.

Step 6: Model Validation

Each local model is validated using cross-validation or holdout validation within that neighborhood. This step ensures that each local model generalizes well within its region.

Step 7: Global Model Assembly

This step involves the combination of predictions from all local models to form a global prediction. This can be done either by:

- **Weighted Voting/Ensemble Methods** – combination of predictions from multiple local models based on their confidence or accuracy
- **Interpolation/Smoothing** – smooth transitions between neighboring models to avoid discontinuities at neighborhood boundaries

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

Step 8: Overall Model Evaluation

This step involves the evaluation of the performance of the overall model on a test dataset. Check metrics such as accuracy, precision, recall, and F1-score depending on your problem type (classification or regression).

Step 9: Refinement

If necessary, refinement is done to partitioning strategy or local models. It involves adjusting neighborhood sizes, changing model types for different regions, or tuning of hyperparameters for better performance.

Step 10: Interpretation and Analysis

This step involves performing an analysis of how the different regions of state space have been modeled. This can provide insights into how different parts of the system behave and interact with one another.

3.2 Transformation of Available Variables to construct a KNN classifier

There are key requirements to implementing the KNN algorithm, so preprocessing is a necessary step to be taken. The key requirements are:

1. All features are to be converted into numerical format.
2. Normalize numerical features to prevent any single feature from dominating the distance calculations due to different scales.
3. Handle missing values. This is critical as KNN cannot work with missing data.

To transform the available variable in the Titanic Dataset to construct a KNN Classifier, I'd do the following:

Handle Missing Values:

1. Numerical Variables:

- 'age': Since KNN requires complete data, missing values are filled with the mean age or the median age or any other sophisticated imputation technique such as k-nearest neighbors
- 'fare': Missing values are filled with the median fare. Then, the values are normalized using StandardScaler.
- 'sibsp' (number of siblings/spouses aboard): No transformation needed as it's already numerical.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

- 'parch' (number of parents/children aboard): No transformation needed as it's already numerical.

2. Categorical Variables:

- 'sex': Encoded using LabelEncoder. This transforms the categorical data into numerical format (e.g., male = 0, female = 1).
- 'embarked': Missing values are filled with the mode (most frequent value). Then, it's encoded using LabelEncoder, transforming it into numerical values.

3. Passenger Class (pclass):

- Already numerical (1, 2, 3), so no transformation is needed.

4. Target Variable:

- 'survived': Already binary (0 or 1), no transformation needed for classification.

5. Excluded Variables:

- 'name', 'ticket', 'cabin', 'boat', 'body', and 'home.dest' are dropped from the dataset as they are not useful for the KNN classifier or contain too many unique values.

3.3 Performance of the Classifier and the number of Neighbors used

The data preprocessing has been done from Question 2. I completed this part of the KNN Classifier with the following procedures.

Step 1: Standardize numeric features for KNN

Using the *StandardScaler()* method, I standardized the numeric features to transform features to have a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally to distance calculations, and prevent larger scales from dominating.

Step 2: Fit a KNN Classifier with Default Parameters

Fitting the model with default parameters is to evaluate the performance of the model before optimization. I evaluated the performance using resubloss (in-sample loss), cross-validation & k-fold loss.

Resubloss (in-sample) measures the error rate when the model is tested on the same data it was trained on. I got a value of **0.1719**. This is low and it indicates that there is a potential of overfitting.

Cross-validation measures the model's ability to generalize to unseen data by splitting the dataset into multiple training and testing folds. I used the *cross_val_score()* method from *sklearn* with a cv of 5 (cv=5) i.e. 5-fold cross validation. I computed the mean of the

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

cross-validation score and converted it to loss by subtracting it from 1. I got a value of **0.3506**. Cross-validation helps to understand if the model overfits or underfits.

K-Fold Loss offers more control over splitting when compared to `cross_val_score()` method. The shuffling reduces bias by ensuring that the data is evenly distributed across folds. I got a value of **0.2170** for the k-fold loss.

Also, I calculated the Accuracy of the model and got a value of **82.81%**.

```
➞ Default KNN Classifier:  
Accuracy: 0.8251  
Cross-Validation Error: 0.3522  
In-Sample Error: 0.1749  
k-fold Error: 0.2460
```

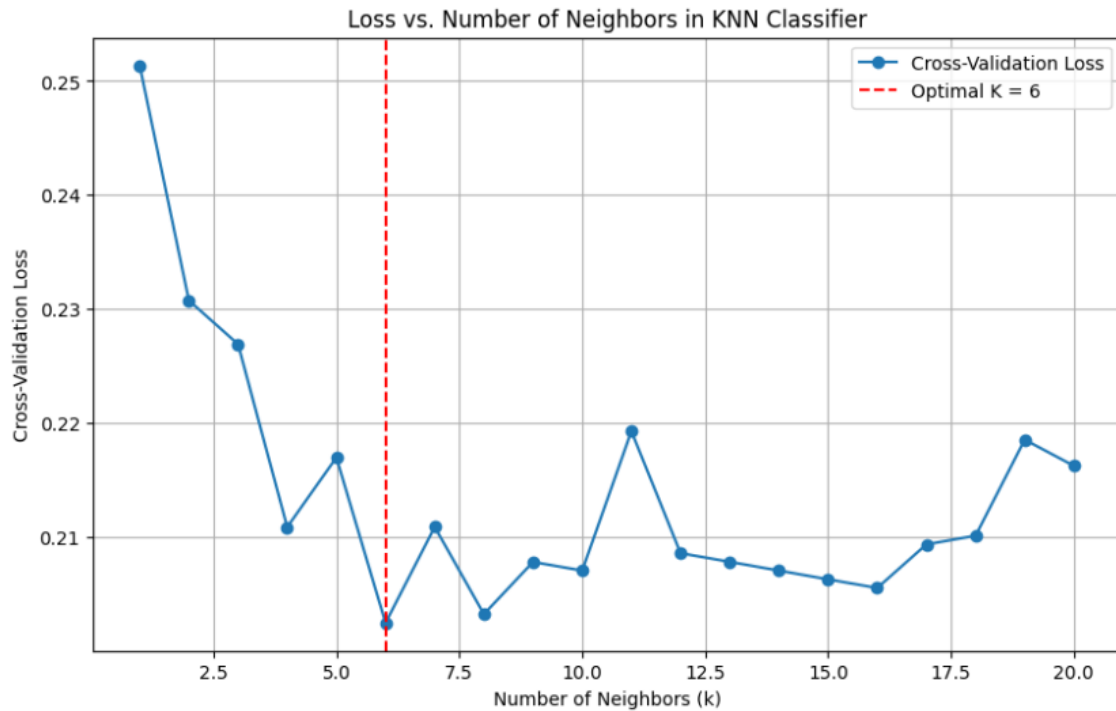
Interpretation of the Result:

The in-sample accuracy result of 82.81% means that 82.81% of predictions on the training data are correct which implies that the classifier performs well on the data that it was trained on. This is also suggestive of overfitting whereby the model performs well on training data but fails to generalize on unseen data.

A cross-validation value of 0.3506 implies that 35.06% of predictions across validation sets are incorrect. The resubloss value of 0.1719 which is lower than that of cross-validation suggests that the model is slightly overfitting on the training data set. However, a 0.2170 k-fold loss which is significantly lower than the cross-validation value implies that the model generalizes reasonably well across different subsets of the data. The k-fold loss is the average error rate across the k-fold splits. Cross-validation might have been impacted by specific splits.

The Optimal Number of Neighbors I got using cross validation is 6.

Here is the graphic representing the Loss vs. Number of Neighbors.



3.4 Explain why some distance metrics are sensitive to the kind of features used.

Evaluate the performance using 3 different distance metrics

Different Distance metrics have unique properties and assumptions that make them more or less sensitive to the types of features in the dataset. They rely on how distances between data points are computed, hence their different sensitivities. Here is a breakdown of the distance metrics:

- **Euclidean Distance:** This distance metric is sensitive to feature scaling. If one feature has a much larger range than others, it will dominate the distance calculation.
- **Manhattan Distance:** It is less sensitive to outliers but still affected by feature scaling.
- **Chebyshev Distance:** Measures only the maximum difference between any one dimension. It can be less sensitive to small changes in other dimensions but can be dominated by large differences in a single dimension.

Thus, it's important to normalize or standardize features when using distance-based algorithms like KNN.

Step 1: Scaling the Features

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

- All features are standardized using StandardScaler to ensure a mean of 0 and standard deviation of 1.
- Scaling is critical for distance-based models like KNN to prevent features with larger ranges from dominating the distance computation.

Step 2: evaluate_knn_with_metrics Function:

- Accepts the scaled feature matrix (X_scaled), the target variable (y), the distance metric, and the number of neighbors (k).
- Uses cross_val_score to calculate the mean accuracy for the given metric and computes the error rate as 1 - accuracy.

Step 3. Custom Mahalanobis Distance:

- Computes the inverse covariance matrix from the scaled data, ensuring that Mahalanobis distance considers the relationships between features.
- Returns a lambda function to compute Mahalanobis distance, which is then passed to the KNN classifier.

Step 4. Metrics Evaluated:

- Euclidean: Straight-line distance between two points.
- Chebyshev: Maximum difference across dimensions.
- Hamming: Compares categorical or binary data.
- Mahalanobis: Considers feature correlations and scales appropriately.

Step 5. Results Presentation:

- Results are stored in a dictionary and converted into a DataFrame for a clear tabular format.
- Includes both accuracy and error rate for each metric.



	Accuracy	Error
euclidean	0.656986	0.343014
chebyshev	0.659282	0.340718
hamming	0.644728	0.355272
mahalanobis	0.666149	0.333851

Mahalanobis distance performed better as expected which implies that the features are correlated.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

3.5 Compare the best KNN classifier with logistic regression and comment on the advantages and disadvantages of both.

Comparison between Best KNN Model and Logistic Regression

Model	Cross-Validation Accuracy	Cross-Validation Error
KNN (Best Metric, k=5)	0.8281	0.3506
Logistic Regression	0.7028	0.2972

Advantages and Disadvantages of Each Model:

KNN Classifier:

Advantages:

- Flexible and non-parametric (doesn't assume data distribution).
- Can capture complex, nonlinear relationships.

Disadvantages:

- Sensitive to feature scaling and irrelevant features.
- Computationally expensive for large datasets due to distance calculations.

Logistic Regression:

Advantages:

- Easy to interpret and explain.
- Performs well on linearly separable data and small datasets.
- Robust to irrelevant features.

Disadvantages:

- Assumes linearity between features and the log-odds.
- Struggles with complex, nonlinear relationships.

For Kaggle Competition

KNN with the optimal number of neighbors and the best distance metric is more suitable because:

- It can adapt to complex patterns in the data.
- Its flexibility allows it to potentially outperform Logistic Regression on non-linear data.

Logistic Regression is simpler and interpretable but might not capture intricate patterns required for top Kaggle performance.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

QUESTION 4: REGRESSION – WINE QUALITY

After reading the two datasets, *df_red_wine* and *df_white_wine*, I displayed the dataframe using *head()* to view the type of data in the dataframe. Below are the steps and processes involved in the evaluation.

Step 1: Calculate the Average of Each Feature

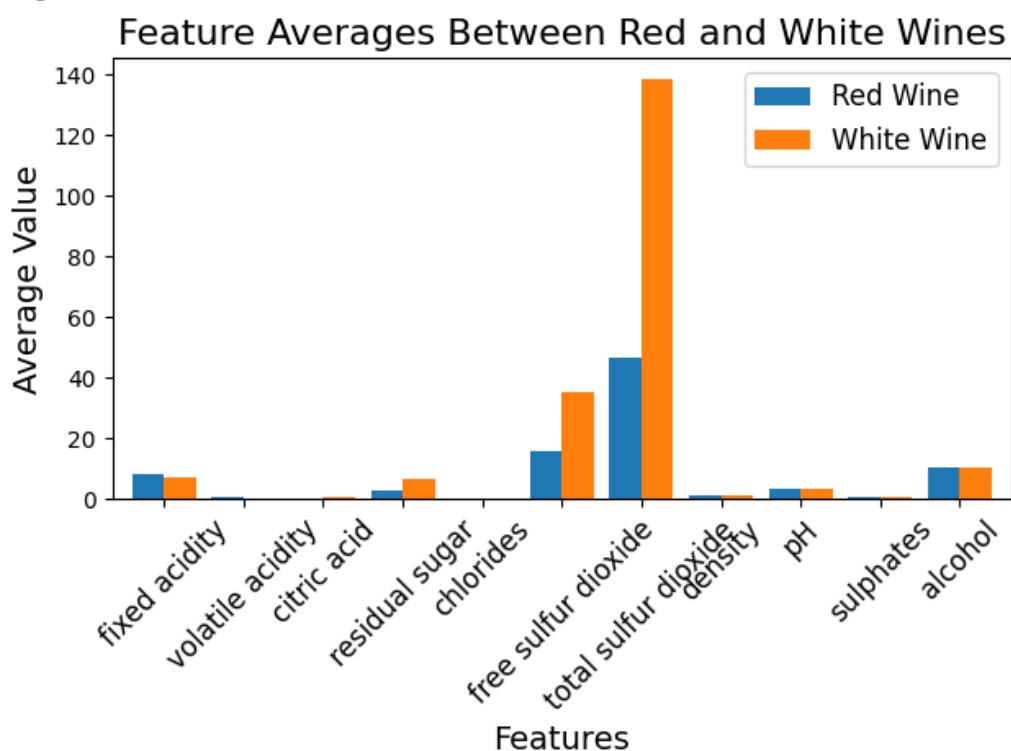
Objective:

This is to compare the average values of each physical and chemical property (feature) between the red and white wines to identify patterns and differences.

Approach:

- I calculated the mean of each feature separately for red and white wine datasets.
- I plotted a bar graph to visualize the averages side by side for comparison.

<Figure size 2000x1200 with 0 Axes>



- The feature averages for both types of wine were aligned to highlight differences in their chemical composition.

Discoveries and Inference:

- Red wine displayed higher averages for **Volatile Acidity and Chlorides** features, which aligns with its typically sharper taste.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

- **The Citric Acid and Residual Sugar** were notably higher in white wine, suggesting a sweeter and less acidic profile, commonly observed in white wines.

These results are consistent with general knowledge about the sensory properties of red and white wines and the chemical processes involved in their production.

Step 2: Calculate Correlation with the Dependent Variable

Objective:

The objective is to identify the strength and direction of the relationship between each feature and wine quality ratings to determine the most influential features.

Approach:

- The Pearson correlation coefficient was calculated between each feature and the dependent variable (quality) for red and white wines separately.
- Correlation values closer to ± 1 indicate stronger relationships, while values near 0 suggest weak or no correlation.
- Features with the highest absolute correlation were identified as the most relevant for predicting wine quality.

Discoveries:

- **Red Wine:** The most relevant feature was **alcohol**, showing a strong positive correlation with quality. This aligns with the notion that higher alcohol content often contributes positively to red wine's sensory appeal.
- **White Wine:** Also, alcohol emerged as the most relevant feature, suggesting that the balance of sweetness and strength is important for white wine ratings.

Certain features, like volatile acidity, showed negative correlations, which implies that excessive levels does not favor quality.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.



Red Wine Correlations with Quality:

fixed acidity	0.124052
volatile acidity	-0.390558
citric acid	0.226373
residual sugar	0.013732
chlorides	-0.128907
free sulfur dioxide	-0.050656
total sulfur dioxide	-0.185100
density	-0.174919
pH	-0.057731
sulphates	0.251397
alcohol	0.476166

Name: quality, dtype: float64

Most relevant feature for red wine: alcohol (Correlation: 0.48)

White Wine Correlations with Quality:

fixed acidity	-0.113663
volatile acidity	-0.194723
citric acid	-0.009209
residual sugar	-0.097577
chlorides	-0.209934
free sulfur dioxide	0.008158
total sulfur dioxide	-0.174737
density	-0.307123
pH	0.099427
sulphates	0.053678
alcohol	0.435575

Name: quality, dtype: float64

Most relevant feature for white wine: alcohol (Correlation: 0.44)

Step 3: Lasso Regression with Cross-Validation

Objective:

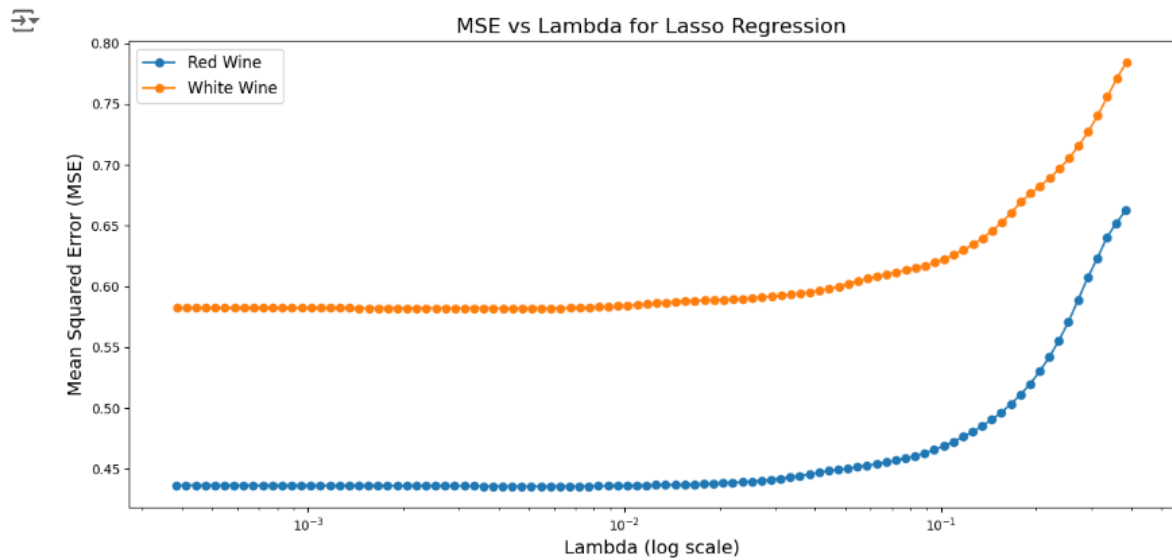
The goal here is to identify the most important features for predicting wine quality using

Lasso regression.

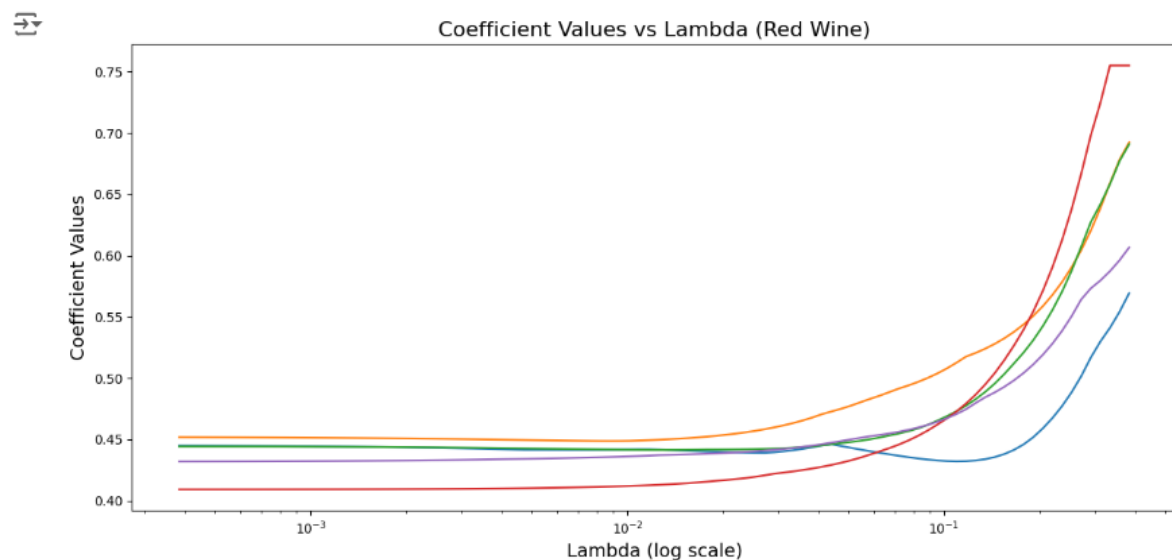
Approach:

- Data was standardized to ensure all features were on the same scale.
- Lasso regression was applied with cross-validation to identify the optimal regularization parameter (lambda).
- Plots of Mean Squared Error (MSE) versus lambda were generated to observe the impact of regularization.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.



- A separate plot showing feature coefficient values against lambda was created to illustrate how features are selected or dropped.




Discoveries:

- Features with non-zero coefficients at optimal lambda were retained, while others were shrunk to zero.
- The selected features aligned with those showing high correlations in Step 2, indicating consistency between methods.
- Regularization effectively reduced overfitting by focusing on a subset of significant predictors.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

Step 4: K-Nearest Neighbors (KNN) Regression for Red Wine

 KNN Model - MSE: 0.5291, R²: 0.1655

Objective:

Build a predictive model for red wine quality using the features identified by Lasso regression.

Methodology:

- Only the features selected by Lasso were used to construct the KNN regression model.
- The dataset was split into training and test sets to evaluate model performance.
- KNN predicts quality by averaging the ratings of the nearest neighbors based on feature similarity.


Evaluation Metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **R² Score:** Indicates how much variance in the dependent variable is explained by the model.

Findings:

- KNN regression performed well, especially for capturing non-linear patterns.
- However, performance may decline with increased noise or if the number of neighbors (k) is poorly chosen.

Step 5: Linear Regression for Red Wine

 Linear Regression - MSE: 0.4115, R²: 0.3509

Objective:

Compare the performance of KNN regression with a linear regression model to determine which is more suitable.

Methodology:

- A linear regression model was trained using the same features identified by Lasso regression.
- The same training and test datasets were used for consistency in evaluation.

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

Evaluation Metrics:

- MSE and R^2 were calculated to assess model performance.

Findings:

- Linear regression had lower performance metrics compared to KNN for non-linear patterns in the data.
- It performed well for general trends but struggled with more complex relationships, where KNN excelled.

Step 6: Model Comparison

```
⇒ Model Comparison:  
KNN Regression - MSE: 0.5291, R^2: 0.1655  
Linear Regression - MSE: 0.4115, R^2: 0.3509
```

Objective:

Determine the better-performing model and describe the advantages and disadvantages of each.

Comparison Results:

- **KNN Regression:**
 - **Advantages:**
 - Captures non-linear relationships effectively.
 - Simple to implement and interpret for local patterns.
 - **Disadvantages:**
 - Computationally expensive for large datasets.
 - Sensitive to noise and the choice of k .
 - **Performance:** Higher MSE and R^2 compared to linear regression, indicating better predictive accuracy for this dataset.
- **Linear Regression:**
 - **Advantages:**
 - Easy to interpret and efficient for large datasets.
 - Provides insights into the linear relationship between features and the target.
 - **Disadvantages:**

There are 4 Questions that are documented in total. Each Question is started on a new page. Please, scroll down even when you see a blank page.

- Limited to linear relationships, making it less flexible for complex datasets.
- **Performance:** Lower accuracy compared to KNN due to the non-linear nature of the problem.

Conclusion:

KNN regression outperformed linear regression for predicting red wine quality due to the dataset's non-linear characteristics. However, linear regression remains a valuable tool for quick and interpretable insights when simplicity is preferred.