Problem 1: Types of Attributes

Classify the following attributes as nominal, ordinal, interval, ratio. Explain why.

(a) Rating of an Amazon product by a person on a scale of 1 to 5: Ordinal

While there is a clear order (5 is better than 4, which is better than 3, etc.), the differences between ratings are not necessarily equal. The difference between a 4-star and a 5-star rating may not be the same as the difference between a 2-star and a 3-star rating in terms of customer satisfaction.

(b) The Internet Speed: Ratio

This is a ratio scale because it has a true zero point (0 Mbps means no internet speed) and the intervals between values are equal and meaningful. For example, 100 Mbps is twice as fast as 50 Mbps.

(c) Number of customers in a store: Ratio

It has a true zero point (0 customers means the store is empty) and the intervals between values are equal and meaningful. For instance, 20 customers are twice as many as 10 customers.

(d) UCF Student ID: Nominal

UCF IDs are unique identifiers assigned to students without any inherent order or numerical significance. They are used solely for identification purposes.

(e) Distance: Ratio

Distance has a true zero point (0 distance means no separation) and the intervals between values are equal and meaningful. For example, 10 kilometers is twice as far as 5 kilometers.

(f) Letter grade (A, B, C, D): Ordinal

Letter grades have a clear order (A is better than B, which is better than C, etc.), but the intervals between grades are not necessarily equal. The difference in performance between an A and a B may not be the same as the difference between a C and a D.

(g) The temperature at Orlando: Interval

Temperature scales like Celsius or Fahrenheit have equal intervals between values, but lack a true zero point. Zero degrees does not mean "no temperature," and negative temperatures exist. While you can perform addition and subtraction with temperatures, ratios are not meaningful (20°C is not twice as hot as 10°C).

Problem 2: Exploring Data Preprocessing Techniques

Link to code - https://github.com/shashnavad/Data_Mining/blob/main/HW1/HW1-Changes-ShashankNavad-v1.ipynb

Q 1) The original document titled "titanic-data-science-solutions.ipynb" is a comprehensive data science workflow for the Titanic survival prediction problem. It covers various stages including data acquisition, analysis, feature engineering, and model building. The document provides a detailed approach to solving the Titanic competition on Kaggle, demonstrating data preprocessing, feature creation, and the application of multiple machine learning algorithms.

I was able to reproduce the following accuracies.

Algorithm	**Accuracy (%)**
Random Forest	86.76
Decision Tree	86.76
KNN	84.85
Logistic Regression	80.36
Support Vector Machines	78.23
Linear SVC	79.01
Naive Bayes	72.28
Perceptron	78.34
Stochastic Gradient Descent	74.07

The Random Forest and Decision Tree models performed the best in the original document with an accuracy of 86.76%, while the Naive Bayes model had the lowest accuracy at 72.28%. These results demonstrate the effectiveness of ensemble methods like Random

Forest for this particular dataset. The document includes detailed data preprocessing, feature engineering, and model training steps that led to these accuracy scores.

Data Preprocessing

- The document starts by loading the training and test datasets using pandas.
- It analyzes the available features and their types (categorical vs numerical).
- Missing values are addressed, particularly in the 'Age' and 'Embarked' columns.
- Categorical variables like 'Sex' and 'Embarked' are converted to numerical values.

Feature Engineering

- A new feature 'Title' is extracted from the 'Name' column, grouping rare titles.
- 'FamilySize' is created by combining 'SibSp' and 'Parch'.
- 'IsAlone' is derived from 'FamilySize'.
- 'Age' is grouped into bands using pd.cut().
- 'Fare' is also categorized into bands.
- A new feature 'Age*Class' is created by multiplying 'Age' and 'Pclass'.

Q 2) In my improved data preprocessing approach, I implemented several enhancements to extract more meaningful features and handle missing data more effectively. Here's a summary of my key improvements:

1) Feature Engineering

I created new features to capture family dynamics, including 'FamilySize' and 'IsAlone'. I also extracted titles from names and grouped rare titles, mapping them to numerical values. I introduced 'TicketPrefix' and 'TicketLen' features from the 'Ticket' column. Additionally, I created 'AgeBand' and 'FareBand' features using pd.cut() and pd.qcut() respectively.

2) Advanced Imputation

I used KNNImputer for more sophisticated handling of missing values in 'Age' and 'Fare' columns, replacing the simpler imputation methods used in the original document before.

3) Feature Scaling

I applied StandardScaler to normalize numeric features: 'Age', 'Fare', 'FamilySize', and 'TicketLen'.

4) Categorical Encoding

I used pd.Categorical().codes for encoding categorical variables, including the newly created features.

5) Feature Selection

I implemented SelectKBest with f_classif to choose the most relevant features.

6) Cross-Validation

I utilized cross-validation techniques to ensure more robust model evaluation. Specifically, I implemented StratifiedKFold cross-validation, which helped in obtaining a more reliable estimate of my models' performance across different subsets of the data. This approach reduces the risk of overfitting and provides a more accurate assessment of how well the models generalize to unseen data.

7) Ensemble model

I implemented an ensemble model using Random Forest, XGBoost, and LightGBM classifiers. This approach combines the strengths of multiple algorithms to create a more robust and accurate prediction. The ensemble method I used is a voting classifier, which takes the predictions from each model and determines the final prediction based on the majority vote.

To optimize the performance of each model in the ensemble, I utilized grid search with cross-validation. This technique allowed me to systematically search through a specified parameter space for each algorithm, finding the best combination of hyperparameters.

8) Improved Accuracies

My improved preprocessing led to significantly better model performance. Here are the accuracies I achieved for different models:

Model	My Accuracy (%)	Original Accuracy (%)
Random Forest	98.65	86.76
Decision Tree	98.65	86.76
KNN	87.54	84.85

Logistic Regression	81.82	80.36
Support Vector Machines	81.26	78.23
Linear SVC	81.14	79.01
Naive Bayes	78.79	72.28
Perceptron	78.68	78.34
Stochastic Gradient Decent	66.78	74.07

My preprocessing improvements led to substantial accuracy gains, particularly for the Random Forest and Decision Tree models, which both saw an increase of nearly 12 percentage points. Most other models also showed improvements, with only the Stochastic Gradient Descent classifier showing a decrease in accuracy.

Problem 3: Distance/Similarity Measures

To group boxes based on their shapes (length-width ratio) or size, we would choose different proximity measures:

Grouping Boxes by Shape (Length-Width Ratio)

For grouping boxes based on their shapes, we would use correlation distance as the proximity measure.

Correlation distance is more appropriate in this case because:

1. It focuses on the pattern or relationship between variables rather than their absolute values.

2. The length-width ratio represents a pattern or proportion, which is captured well by correlation.

3. Correlation distance is scale-invariant, meaning it won't be affected by the overall size of the boxes, only their proportions.

For example, two boxes with dimensions 2x1 and 4x2 would be considered similar in shape using correlation distance, despite their different sizes, because they have the same length-width ratio.

Grouping Boxes by Size

For grouping boxes based on their size, we would use Euclidean distance as the proximity measure.

Euclidean distance is more suitable here because:

1. It takes into account the absolute magnitude of the measurements.
2. Size is an absolute property that directly relates to the dimensions of the box.
3. Euclidean distance preserves the scale of the original data, which is crucial when comparing sizes.

For instance, when comparing box sizes, the actual measurements (length, width, height) matter, and Euclidean distance would accurately reflect the differences in these dimensions.

In summary, correlation distance is better for capturing shape similarities based on proportions, while Euclidean distance is preferable for comparing absolute sizes of boxes.

Link to code (Question 2) -
https://github.com/shashnavad/Data_Mining/blob/main/HW1/HW1-Changes-ShashankNavad-v1.ipynb