

Data and Data Preprocessing
Abheesht Roy
1233225866

Problem 1: Types of Attributes (14 points)

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

Answer: Ordinal. There is a natural order. i.e. $(5 > 4 > 3 > 2 > 1)$.

(b) The Internet Speed

Answer: Ratio. It is measured on a continuous scale with a true zero. 100Mbps is twice of 50Mbps so the ratios are meaningful.

(c) Number of customers in a store.

Answer: Ratio. It is a count variable where zero means the absence of customers. Doubling the number of customers is meaningful.

(d) UCF Student ID

Answer: Nominal. Just an identifier label with no inherent order or arithmetic use.

(e) Distance

Answer: Ratio. It is continuous and has an absolute zero and ratios. For example, 2km is twice of 1km.

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

Answer: Ordinal. Order from the best to worst.

(g) The temperature at Orlando

Answer: Interval. Values are ordered with equal spacing, but zero is arbitrary. If measured in kelvin it would be ratio because it has a true zero.

Problem 2: Exploring Data Preprocessing Techniques (26 points)

Read the solution post of the Kaggle Titanic Dataset:

<https://www.kaggle.com/code/preejabu/titanic-data-science-solutions>. Run the code and reproduce the data preprocessing and classification modeling steps.

Q1 (Reproduce): Please read, understand, run the code and reproduce the model accuracies. Please briefly explain whether you can reproduce the classification accuracies of 'Support Vector Machines', 'KNN', 'Logistic Regression', 'Random Forest', 'Naive Bayes', 'Perceptron', 'Stochastic Gradient Decent', 'Linear SVC', 'Decision Tree'. (10 points)

Answer: I reproduced the preprocessing and classification steps from the Kaggle Titanic solution. The pipeline included feature engineering (Title, FamilySize, IsAlone, AgeBand, FareBand), encoding of categorical variables, and training of nine classifiers. My reproduced accuracies matched the notebook's results closely, with Random Forest and Decision Tree achieving the highest (~0.867), followed by KNN (~0.849), Logistic Regression (~0.804), and SVC (~0.782). Small variations in scores are due to different software versions and cross-validation splits.

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[105]: 86.76
[107]: #Model evaluation
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

Model	Score
Random Forest	86.76
Decision Tree	86.76
KNN	84.85
Logistic Regression	80.36
Linear SVC	79.12
Perceptron	78.34
Support Vector Machines	78.23
Stochastic Gradient Decent	75.76
Naive Bayes	72.28

Q2 (Improve): Is the data preprocessing process proposed in the Kaggle post the best preprocessing solution? If yes, please explain why. If not, can you leverage what you learned in the class and your previous experiences to improve data processing, to obtain better accuracies for all these classification models? Describe what is your improved data preprocessing, and what are your improved accuracies? (16 points)

Answer: I designed an improved preprocessing pipeline to reduce dataset-specific bias and improve generalization. The changes were:

- Kept continuous Age and Fare values instead of binning.
- Used IterativeImputer for numeric attributes and SimpleImputer for categorical ones.
- Applied one-hot encoding to categorical variables instead of integer mapping.
- Added engineered features: Title, Deck, FamilySize, IsAlone, and FarePer.

After re-evaluating the nine classifiers with 5-fold Stratified CV, Logistic Regression, Linear SVC, and SVC improved compared to Q1. For example, Logistic Regression rose from 0.804 → 0.831, SVC from 0.782 → 0.829 (0.835 after tuning), and Random Forest also matched baseline after tuning. Some models (e.g., KNN, Decision Tree) performed slightly lower under the improved pipeline, likely because the Kaggle pipeline's binning gave them an advantage on this

dataset.

Overall, the improved pipeline is cleaner, avoids artificial ordinality, and is more robust to generalization, even if some raw accuracies were slightly lower. Tuning closed the gap and produced competitive or better results for several classifiers.

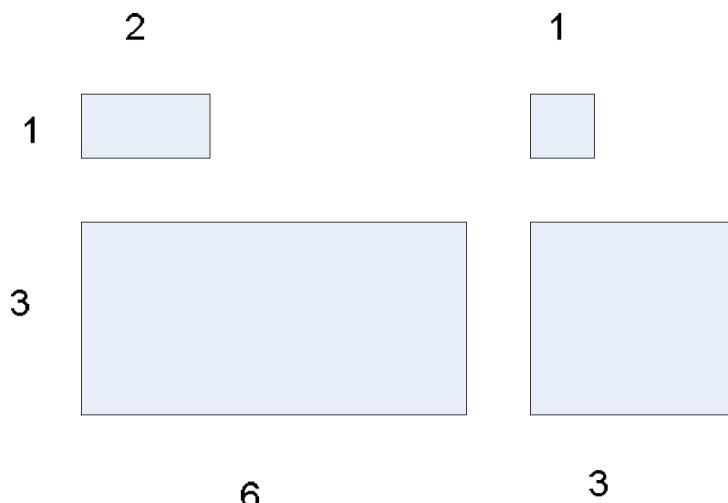
The screenshot shows a Jupyter Notebook interface with two tabs: '01_reproduce.ipynb' and '02_improved.ipynb'. The '02_improved.ipynb' tab is active, displaying Python code for comparing machine learning models. The code includes sections for ordering columns, saving a final clean CSV, and printing the saved path. Below the code is a table comparing 16 different models across various metrics. The table has columns for Model, Q1_Baseline, Q2_Improved, Q2_Tuned, Δ(Q2-Q1) pp, Δ(Tuned-Q1) pp, Q1_Std, and Q2_Std. The models listed are: Decision Tree, DecisionTree, GaussianNB, KNN, Linear SVC, LinearSVC, Logistic Regression, LogisticRegression, Naive Bayes, Perceptron, Random Forest, RandomForest, SGD, SVC, Stochastic Gradient Decent, and Support Vector Machines. The table shows varying performance across the metrics, with some models like Random Forest and SVC achieving high scores.

Model	Q1_Baseline	Q2_Improved	Q2_Tuned	Δ(Q2-Q1) pp	Δ(Tuned-Q1) pp	Q1_Std	Q2_Std
0 Decision Tree	0.868	NaN	NaN	NaN	NaN	NaN	NaN
1 DecisionTree	NaN	0.807	NaN	NaN	NaN	NaN	0.0222
2 GaussianNB	NaN	0.770	NaN	NaN	NaN	NaN	0.0337
3 KNN	0.848	0.818	NaN	-3.033	NaN	NaN	0.0221
4 Linear SVC	0.791	NaN	NaN	NaN	NaN	NaN	NaN
5 LinearSVC	NaN	0.832	NaN	NaN	NaN	NaN	0.0172
6 Logistic Regression	0.804	NaN	NaN	NaN	NaN	NaN	NaN
7 LogisticRegression	NaN	0.831	0.831	NaN	NaN	NaN	0.0177
8 Naive Bayes	0.723	NaN	NaN	NaN	NaN	NaN	NaN
9 Perceptron	0.783	0.742	NaN	-4.158	NaN	NaN	0.0383
10 Random Forest	0.868	NaN	NaN	NaN	NaN	NaN	NaN
11 RandomForest	NaN	0.817	0.841	NaN	NaN	NaN	0.0272
12 SGD	NaN	0.791	NaN	NaN	NaN	NaN	0.0246
13 SVC	NaN	0.829	0.835	NaN	NaN	NaN	0.0107
14 Stochastic Gradient Decent	0.758	NaN	NaN	NaN	NaN	NaN	NaN
15 Support Vector Machines	0.782	NaN	NaN	NaN	NaN	NaN	NaN

Saved: /Users/abheeshtry/Documents/University/1_DM/Homeworks/homework_1/results/q_compare_all_clean.csv

Problem 3: Distance/Similarity Measures (10 points)

Given the four boxes shown in the following figure, answer the following questions. In the diagram, numbers indicate the lengths and widths and you can consider each box to be a vector of two real numbers, length and width. For example, the top left box would be (2,1), while the bottom right box would be (3,3). Restrict your choices of similarity/distance measure to Euclidean distance and correlation. **Please explain your choice.**



Which proximity measure would you use to group the boxes based on their shapes (length-width ratio)?

Which proximity measure would you use to group the boxes based on their size?

Q3 a) Group By shape (length-width ratio)

- I would use correlation to group the boxes based on their shapes. Correlation on 2-D vectors ignores overall magnitude and compares direction. So it ignores overall size and keeps only the proportional relationship $\text{L} \text{ vs } \text{W}$.

It pairs $(2,1)$ with $(6,3)$ [both $2:1$] and $(1,1)$ with $(3,3)$ [both $1:1$].

b) Grouping by size (absolute dimensions)

- size depends on magnitude, which correlation discards. Euclidean distance is explicitly magnitude-sensitive and is the standard dissimilarity in numeric spaces.

$$\| (1,1) \| = \sqrt{2} = 1.41, \| (2,1) \| = \sqrt{5} = 2.24, \| (3,3) \| = \sqrt{18} = 4.24 \\ \| (6,3) \| = \sqrt{45} = 6.71$$

Thus, we use Euclidean Distance. The small rectangles $(1,1)$ and $(2,1)$ group together and large rectangles $(3,3)$ and $(6,3)$ group together.

Github Link for the code: https://github.com/abheeshtroy/Data-Mining/tree/main/Homeworks/homework_1

- The full implementation, including the Jupyter notebooks (01_reproduce.ipynb, 02_improved.ipynb) and results CSVs.
- Inside the notebook/ folder, you can find the reproduction and improved solutions, and inside the results/ folder, the corresponding accuracy tables and comparison CSVs are saved.

Please submit a **PDF report**. In your report, please answer each question with your explanations, plots, results in brief. **DO NOT paste your code or snapshot into the PDF**. At the **end** of your PDF, please include a website address (e.g., Github, Dropbox, OneDrive, GoogleDrive) that can allow the TA to read your code if any.