1 Alejandro Helmrich

Below, you will find a detailed explanation of the code I submitted on blackboard. Most of the comments will have a sequential order to them. Thank you for your attention and sorry if it looks messy.

**Task 1**

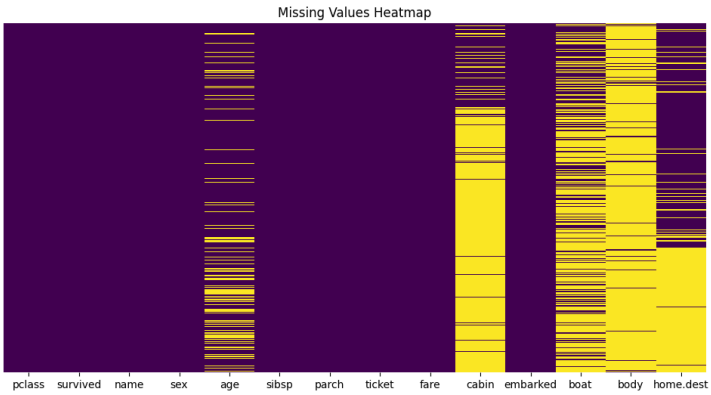
1. Started off by loading the data from the excel, which reads the titanic dataset.
2. Did a basic dataset overview
    a. df.head() -> View the first few rows
    b. df.info() -> See data types and non-null counts
    c. df.describe() -> Get summary statistics for numerical columns
3. Did a short EDA (Exploratory Data Analysis) where I tried to:
    a. Count plots help visualize the distribution of categorical variables like survival, class, and gender
    b. Shows the age distribution in the histogram, where it gives insights into the range and central tendency
    c. Make a correlation heatmap (using only numeric columns) that helped identify relationships between variables (for example, if two features move together)

**Task 2**

1. Here I started off by calculating the number of missing entries per column and only printing those with missing data:

```
age          263
fare           1
cabin       1014
embarked       2
boat         823
body        1188
home.dest    564
dtype: int64
```

2. After that I visualized the data using a heatmap that shows where missing data exists in each of the columns (yellow bars) while also showing the existing data (purple)




3. My strategy on missing values:

a. Age: Represent variable using the median/mean based on the class or gender of the passengers
b. Cabin: Might drop it as there are a lot of missing values
c. Boat: As boat is the assigned lifeboat to each individual, it is best to mark as 'None' since this may just be because they did not assign all of the people one
d. Body: Mark as 'Not Found' since the missing data are most likely the missing bodies.
e. Home.dest: drop; not relevant as it might lead to un-insightfull conclusions

This strategy is essential because many ML algorithms cannot handle missing values directly.

**Task 3**

1. I started off by filling missing categorical values with 'Unknown' and then, the columns were selected for encoding. The values seen in the code ('pclass', 'sex', 'embarked') were chosen as they are usually considered good predictors for overall survival (I asked gpt about this).
2. In terms of encoding the data, I absorbed the following:
    a. The dataset shape changed from (1309, 14) to (1309,20) as each categorical column was split into multiple encoded columns.

3. Impact on ML:
    a. Numeric format so algorithms that require numeric inputs can handle it
    b. No normal bias -> each column is separate for each category so the model won't assume a correlation between categories
    c. Allows for more complexity in models
    d. Better interpretability -> categories become its own binary feature -> easy to determine which category influences model

**Task 4**

Difference

- Standardization
    - Centers data around a mean of 0 and standard deviation of 1 (with values being both positive and negative (above or below mean))
    - Used in algos that assume data is normally distributed or sensitive to feature scale
- Normalization
    - Rescales features to a [0,1] range
    - Used in algos sensitive to magnitued of values or requiring a bound

Importance

- Prevents one feature with a large range from dominating the ml Model
- Faster optimization
- Essential for distance based algorithms (clustering, KNN) and gradient based methods (neural networks/logistic regressions) (got these examples from GPT)

**Task 5**

Here I decided to split the data into the following:

60% training

20% validation
20% testing

1. Through stratification by the variable we can ensure that class distribution (ppl surviving vs not surviving) is maintained across the 3 splits
2. In the code I also chose to drop irrelevant columns -> name, ticket, etc.. -> as it was best to focus on features that make the model more predictive

**Task 6**
Here I started off by preprocessing the training set -> hot encoded categorical columns so that features maintained numeric and then I added missing values into the numeric columns by taking the median and adding it to the missing data.

After that I used SMOTE to balance the training set -> unsampled minority class
-> helps with bias (towards majority class) evasion

**Task 7**
In this task I focused on dropping the low variance and highly correlated features (as you instructed)

Why did we perform this only on a training set? -> We perform tasks 6 and 7 (handling class imbalance and feature selection) only on the training set to prevent data leakage. If we applied them to the entire dataset, information from the validation or test sets would influence our preprocessing, leading to overly optimistic performance estimates and reduced generalizability.

**Task 8**
Started off by pre-processing (one-hot encoding, imputation and feature selection) the validation set -> make sure they match that of the training set.

Then I chose linear regression as the model for the logistic regression. It has a good interpretability and it is a common baseline classifier (according to gpt, hence I choose it). This model was also trained based off the training set.

Visual analysis:
**Confusion Matrix:** Visualizes how many predictions were correct versus incorrect (true positives, true negatives, false positives, false negatives).
**Accuracy and Classification Report:** Provide overall performance metrics (accuracy, precision, recall, F1-score) for each class.
**ROC Curve:** Plots the trade-off between the true positive rate and false positive rate at various threshold settings; the Area Under the Curve (AUC) indicates discriminative power.
**Precision-Recall Curve:** Particularly useful in imbalanced scenarios, it shows how precision changes as recall increases.
**Feature Importance:** Displays model coefficients to show which features increase or decrease the log-odds of survival. This helps in understanding the influence of each feature.