Bader Al-Hilawani
1009724498

Question 1: Data Cleaning

For data cleaning, the original DataFrame was loaded, and a heatmap was plotted to visualize the number of empty columns. The heatmap revealed a substantial number of empty columns, which initially suggested they might be candidates for removal. However, upon closer inspection, it became clear that many of these columns were empty because of the way the data was structured: they represented multiple-choice questions where respondents could select all applicable options. When a respondent did not select a particular option, it appeared as blank, resulting in "NaN" values. Consequently, not all of these columns should be dropped, as encoding techniques like multi-encoding can effectively handle this structure.

The first step in the data cleaning process was to address errors in column 'Q26' in the .CSV file, which prevented it from accurately displaying information about the number of individuals responsible for data science workloads at each respondent's workplace. Once this issue was resolved, a few columns were dropped: the first row (which only contained question details), the first column (which recorded survey completion time and was not relevant to predicting salary), column 'Q5' (as it had the same response for all entries, adding no valuable information), and column 'Q29' (which was the target variable). Additionally, columns with more than 95% missing values were dropped to reduce the dataset's sparsity. After these steps, the number of features was reduced from 298 to 163.

Next, the missing values in single-column responses were addressed. These columns, identified by the absence of an underscore (_), were filled with the mode within each salary bucket for better accuracy. Once the missing values were imputed, the single-column responses were encoded: ordinal encoding was applied to columns with range-based values, while one-hot encoding was used for the rest.

For multi-column responses, each cell with a selected option was replaced with [1], and each blank cell was replaced with [0], where [1] indicates the user selected the option, and [0] indicates they did not.

Question 2: Exploratory data analysis and feature selection:

For visualization, only the top 20 features were plotted to highlight the most influential factors. This graph can be found in **Appendix-1**

For feature engineering, two new features were created: age * years of coding and age * years of machine learning experience. These additions integrate age with coding and machine learning experience, potentially enhancing salary prediction by capturing the combined effect of age and expertise in these areas.

For feature selection, logistic regression with L1 regularization was applied to identify the most important features. A grid search over various C values was performed to optimize feature selection, as L1 regularization inherently performs feature selection by zeroing out coefficients of less useful features. As a result, the number of features was reduced from 282 to 197. From these 197 features, the top 100—those with the highest coefficient values—were selected for model training.

Question 3: Model implementation:

Based on the table in **Appendix-2,** the model's accuracy using the default hyperparameters decreases slightly with each cross-validation (CV) fold. The average accuracy across all folds is 87.06%, with a variance of 1.15e-04, indicating a consistent performance and a strong training accuracy. One key hyperparameter that directly impacts the bias-variance trade-off is the C value, which controls regularization strength.

To evaluate how different C values affect the model's bias (error due to simplifying assumptions) and variance (error due to sensitivity to fluctuations in the training data), a grid search was performed. The results, shown in the bias-variance graph found in **Appendix-3**, align with expectations: as regularization strength increases (lower C values), bias increases while variance decreases. The ideal C value, highlighted by the red dotted line, is C = 1, indicating an optimal balance that slightly increases bias while effectively reducing variance.

Scaling/normalization is necessary because logistic regression performs better when features are on a similar scale. Logistic regression is sensitive to feature scales, particularly when regularization is applied, as the scale affects each feature's contribution to the model. Without scaling, L1 or L2 regularization may disproportionately penalize features with larger scales, leading to uneven regularization.

Question 4: Model tuning:

Accuracy alone is often an unsuitable metric in certain scenarios, particularly for ordinal classification tasks or when the dataset is imbalanced:

Ordinal Nature of the Target: In ordinal logistic regression, the target variable has an inherent order (e.g., "low," "medium," "high"). Accuracy alone overlooks this order and treats each class as equally distinct from every other class, which can lead to misleading evaluations. For example, if a model predicts "low" instead of "medium," it should be penalized less than if it predicts "high" instead of "medium." Metrics such as mean absolute error (for ordinal misclassifications) or weighted F1-score better capture the model's ability to understand the ordered relationships.

Imbalanced Classes: When certain classes are more frequent, accuracy may be biased towards predicting the majority class. For instance, if 90% of the data belongs to one class, predicting this majority class for all instances would yield 90% accuracy, even though the model may perform poorly on minority classes. Metrics like precision, recall, or F1-score (either macro or weighted) provide a more balanced evaluation by accounting for performance across each class.

For these reasons, it is more suitable to use precision, recall, F1-score, or a combination of them for a balanced evaluation, or to use mean absolute error for ordinal classifications.

The hyperparameters for ordinal logistic regression are regularization strength (C), penalty type (L1, L2, or Elastic Net), solver, max iterations, tolerance, class weight, random state, and (if applicable) multi-class strategy. The following are key hyperparameters for logistic regression that can be tuned for ordinal logistic regression:

Regularization Strength (C): This parameter controls the strength of regularization, with a smaller C indicating stronger regularization. Adjusting C helps manage the bias-variance trade-off, allowing for tuning of the model's flexibility.

Penalty Type: Options include l1 and l2. The l1 penalty encourages sparsity, potentially reducing the model to a smaller subset of important features, while the l2 penalty encourages smaller coefficients for all features without necessarily setting any to zero. The choice between l1 and l2 affects feature selection and model interpretability.

These two hyperparameters, C and penalty type, were tuned to improve model performance based on an appropriate metric, such as weighted F1-score or mean absolute error.

After tuning, the best hyperparameters and corresponding score are as follows:

- Best hyperparameters: {'C': 10, 'penalty': 'l1'}, Best score: 0.8734

The updated feature importance based on the tuned model shows that the coefficients for some features have increased or decreased, resulting in a new ranking of feature importance which is shown in **Appendix-4**.

Question 5: Testing & Discussion:

Based on the best model, the test and train accuracy is shown in the **Appendix-5**. The model's performance on the training and test sets is very close, with only a small drop in accuracy and F1-score from training to test. This indicates that the model is not significantly overfitting. If it were overfitting, we would expect a much higher training accuracy compared to the test accuracy. Conversely, if it were underfitting, both the training and test accuracies would be lower than what we observe here. For further improvements the c value can be further tuned, more features can be added through feature engineering, and other hyperparameters can be tuned.

From the distribution plots of the true vs. predicted target classes for both the training and test sets (located in **Appendix-6**), we can gain several insights into the dataset and the performance of the trained model. The first being class imbalance. Both the training and test sets show an imbalance in the true target class distribution, with more instances in class 1 than in class 0. This class imbalance is a likely reason why the model is biased towards predicting class 1 more frequently. Model Prediction Bias. The model shows a strong tendency to predict class 1 over class 0 on both the training and test sets, aligning with the imbalance in the data. This suggests that the model has learned to favor the majority class, possibly to achieve higher overall accuracy.

Appendices:

Appendix-1

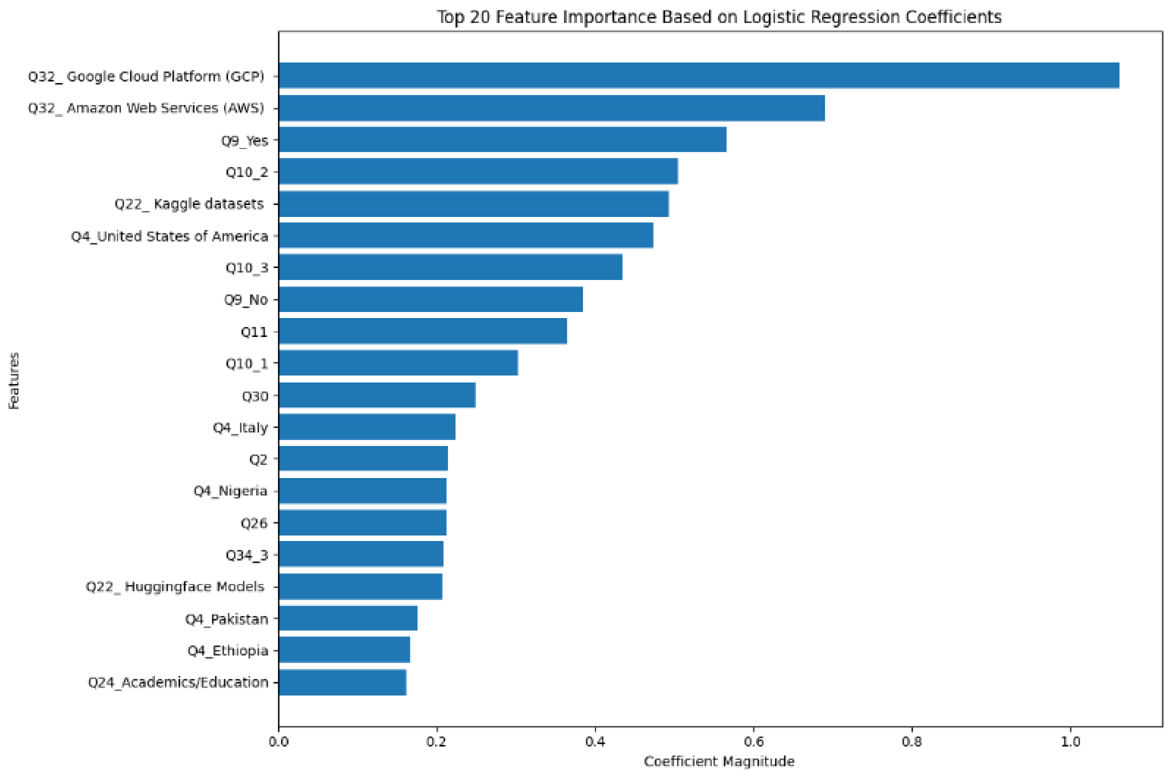

Figure 1: Top 20 features importance based on logistic regression coefficients

Appendix-2

Table 1: 10-Fold Cross-Validation Results with Accuracy for Each Fold

10-Fold Cross-Validation Results:	**Accuracy for each fold:**
Fold 1:	0.8869
Fold 2:	0.8888
Fold 3:	0.8487
Fold 4:	0.8647
Fold 5:	0.8702
Fold 6:	0.8703
Fold 7:	0.8705
Fold 8:	0.8724
Fold 9:	0.8650
Fold 10:	0.8687
Average accuracy across folds:	0.870631908971865
Variance of accuracy across folds:	0.000115253455505364 82

Appendix-3

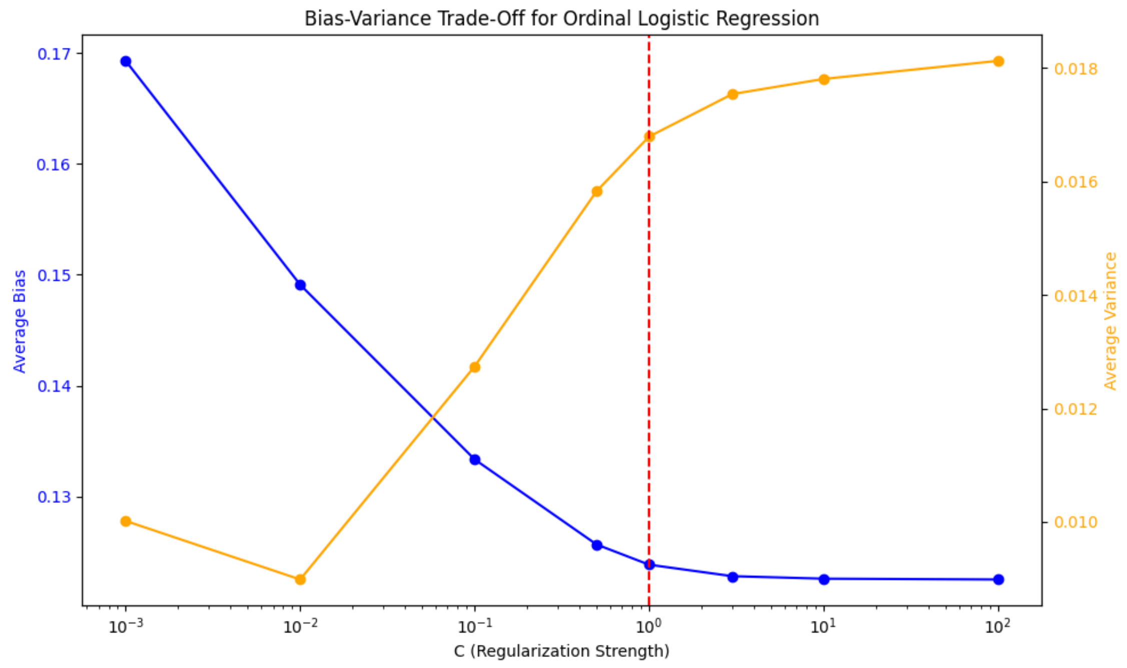

Figure 2: Bias-Variance trade off. Optimal C at 0.1 outlined in red

Appendix-4

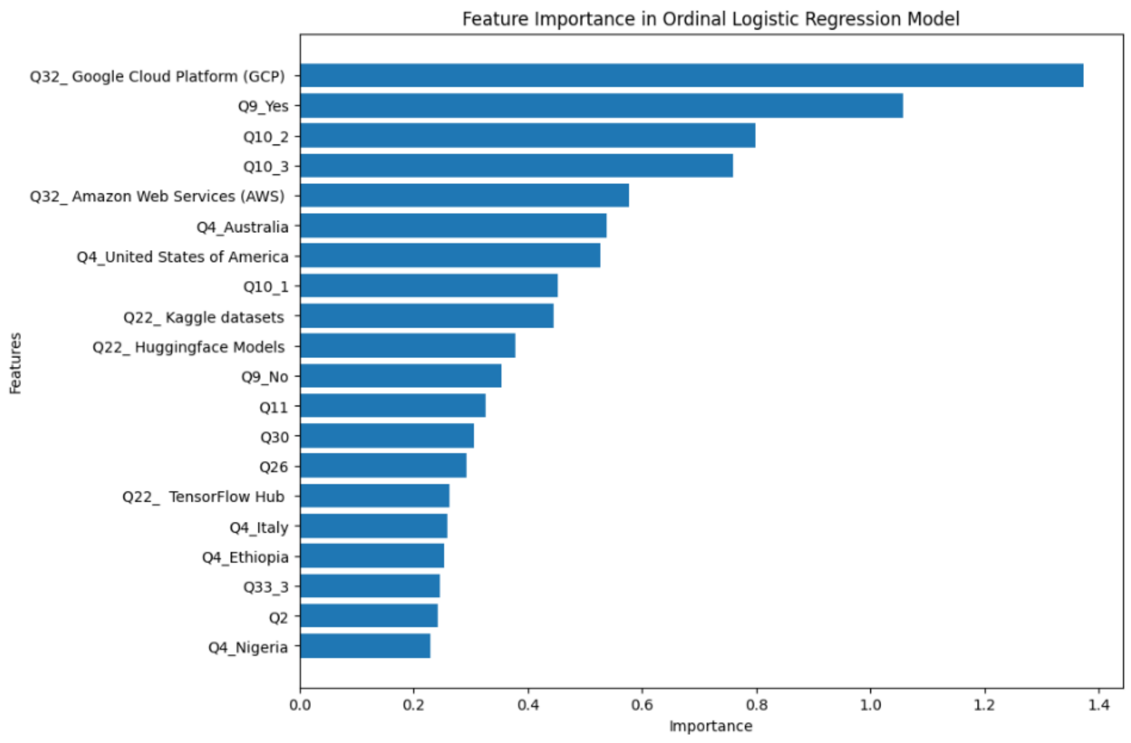

Figure 3: Updated Top 20 features importance based on ordinal logistic regression coefficients

Appendix-5

Table 2: Training and Test accuracy/F1-Score

Training Accuracy:	0.8783143107989464
Training F1-Score:	0.8796632708072172
Test Accuracy:	0.862351495288816
Test F1-Score:	0.8637592911443858

Appendix-6

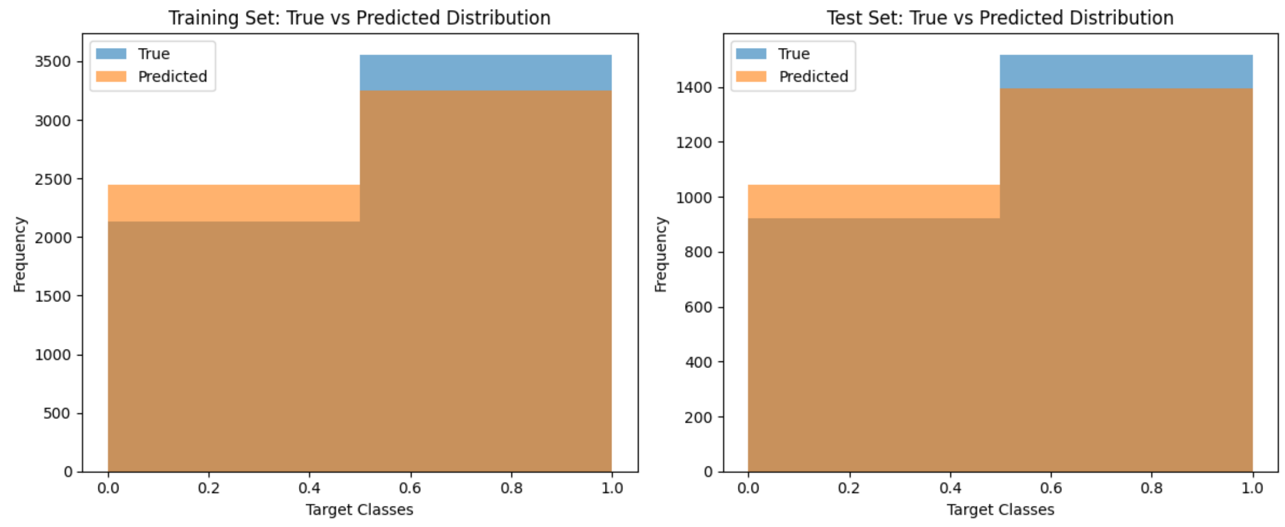

Figure 4: Distribution plots of the true vs. predicted target classes for both the training and test sets