Smoker Status Prediction
(Binary Classification)

Saatvik Sinha **MT2025722**

Table of Contents

1 Introduction **3**

2 Setup **3**

3 Dataset Description **3**

4 Exploratory Data Analysis **4**
- 4.1 Descriptive Statistics . . . 4
- 4.2 Univariate Analysis . . . 4
- 4.3 Bivariate Analysis . . . 5

5 Data Processing Steps **8**

6 Hyperparameter Tuning **10**

7 Comparative Analysis **11**
- 7.1 Interpretation Model-by-Model . . . 12
  - 7.1.1 Logistic Regression . . . 12
  - 7.1.2 Support Vector Classifier (RBF Kernel) . . . 12
  - 7.1.3 Neural Network . . . 13
- 7.2 Cross-Model Interpretation by Preprocessing Strategy . . . 13
  - 7.2.1 Dropping Highly Correlated Features (V2) . . . 13
  - 7.2.2 Removing Outliers (V3) . . . 13
- 7.3 Best Overall Performer . . . 14

8 Future Work **14**

1 Introduction

The experiments were performed in google colab.
The project code ipynb notebooks repository link: https://github.com/Saatvik-Sinha/AIT-511-MT-Project2-SmokerStatus

- The V1 notebook contains training and testing of models without any feature selection or outlier removal.
- The V2 notebook contains training and testing of models after dropping certain features but no outlier removal.
- The V3 notebook contains training and testing models after removing some outliers but without dropping features.

2 Setup

1. Install necessary libraries using pip.
2. Upload kaggle.json file generated from your kaggle account.
3. Import and unzip dataset files from kaggle directly using the JSON file and kaggle library commands.
4. Import all necessary libraries for experimentation.
5. Read the train csv file as dataframe using pandas

3 Dataset Description

This dataset, designed to aid in the development of machine learning models for public health, focuses on predicting an individual's smoking status using a comprehensive set of bio-signals. Motivated by the need to identify smokers who might benefit from cessation assistance beyond traditional, often conflicting indicators like nicotine dependence or carbon monoxide levels, the dataset aggregates health metrics to classify subjects. It includes a variety of physiological features such as age, height, weight, waist circumference, eyesight, hearing, blood pressure (systolic and relaxation), fasting blood sugar, cholesterol levels (Total, HDL, LDL), triglycerides, hemoglobin, urine protein, serum creatinine, and liver enzymes (AST, ALT, GTP), alongside the presence of dental caries, with the primary target variable being the binary smoking status.

4 Exploratory Data Analysis

4.1 Descriptive Statistics

1. The raw dataframe contained 38984 rows and 23 columns.

2. 4 input features and the target feature were categorical, while remaining were continuous.

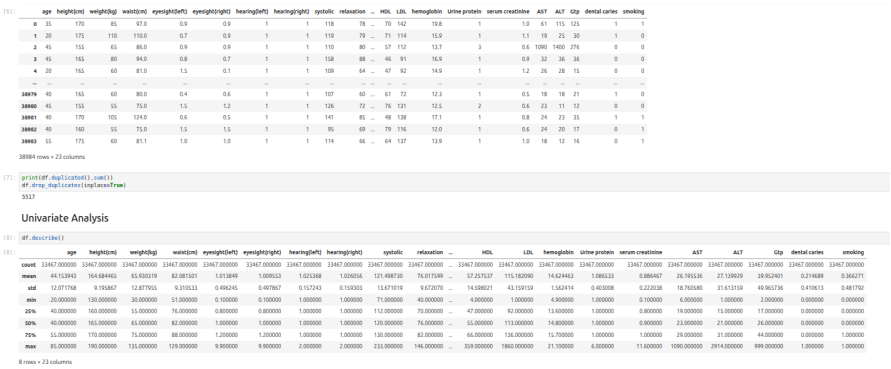
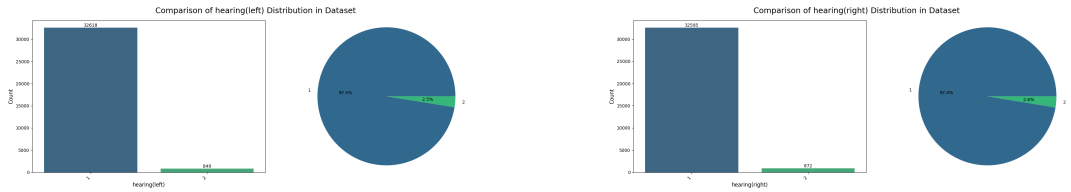


Figure 1: Descriptive Statistics

4.2 Univariate Analysis

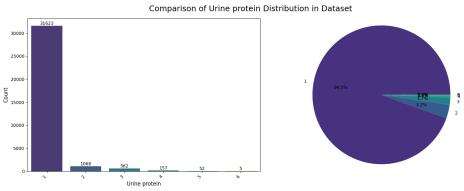
1. There were no null/missing values in the csv file.

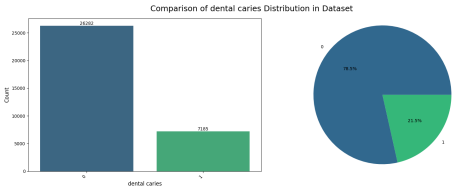
2. All categorical columns and frequencies of their categories were observed as bar graph and pie chart

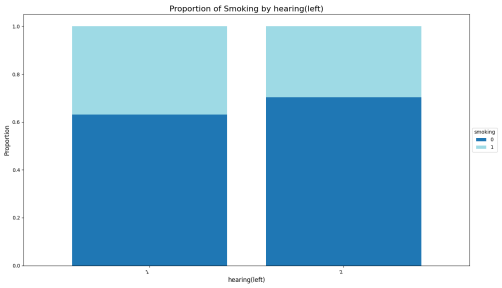


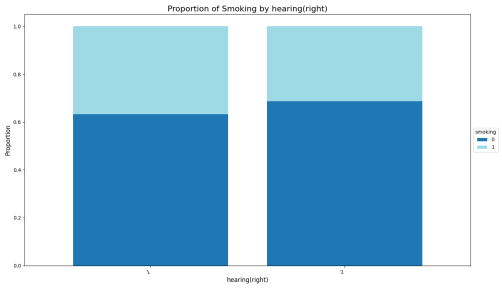
3. Distribution of target categories among categorical features were observed as stacked bar charts to assess their proportion.

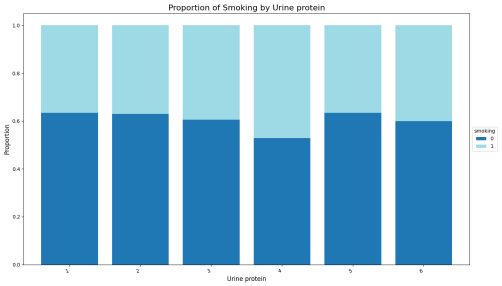
4. Distribution of target categories among continuous features were plotted as histograms and curves to assess likelihood of target value in a given range of continuous values

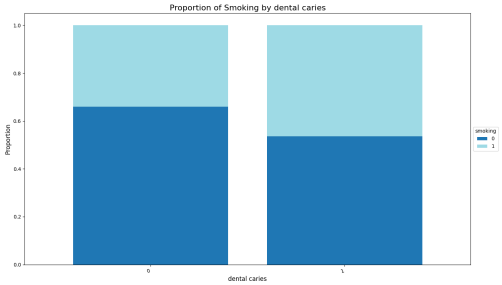












4.3 Bivariate Analysis

1. Correlation Analysis

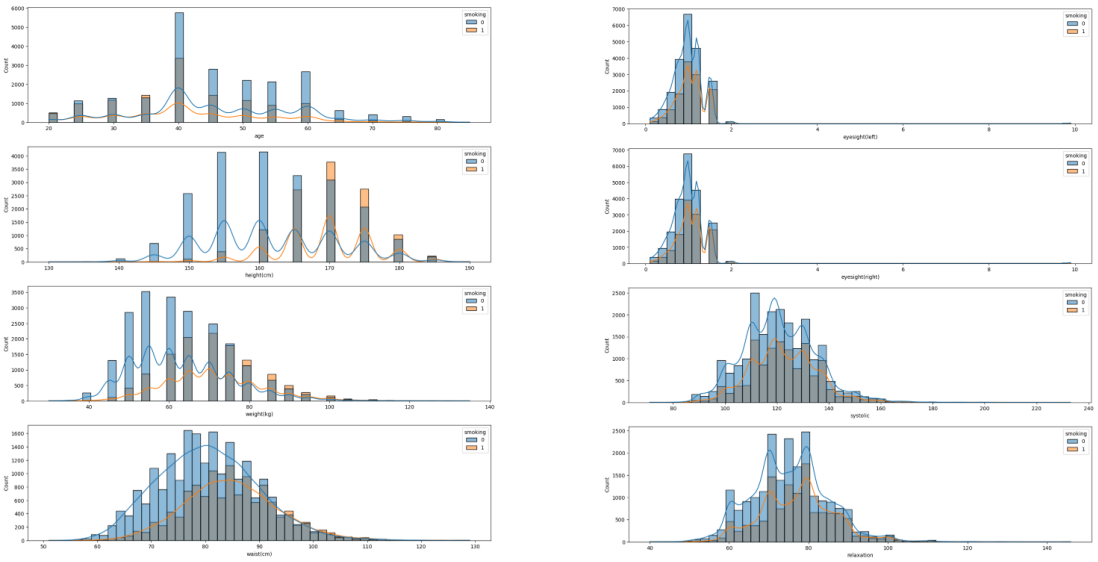
   - It shows that waist and systolic have more than 75% correlation with weight and relaxation respectively, while having lower correlation to smoking compared to their correlated features.

2. Categorical feature vs Categorical feature Analysis

   - The proportion of categories of one categorical feature were checked compared to all other categorical features, separated by target categories. This was observed using stacked bar plots.

3. Categorical feature vs Continuous feature Analysis

   - Box plots were used to detect outlier points



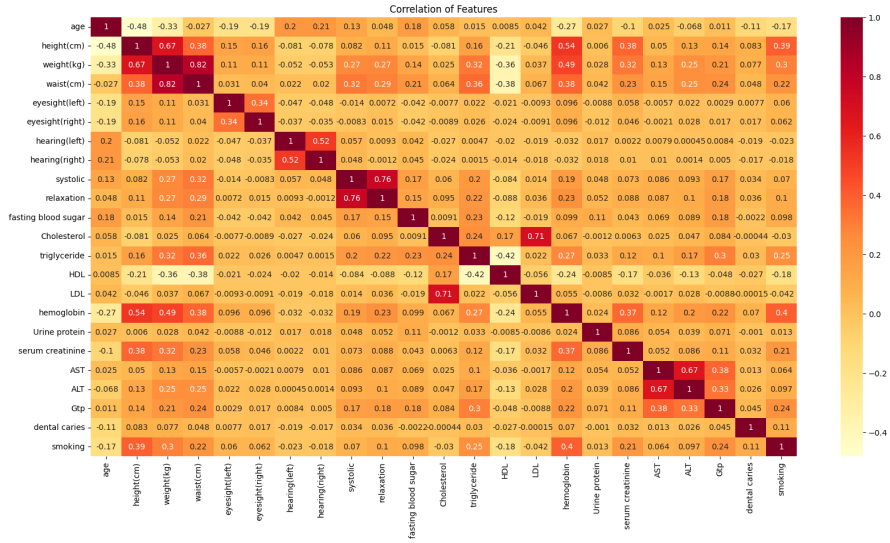


Figure 7: Correlation Matrix

4. Continuous feature vs Continuous feature Analysis

   - It could be observed from the pairplots that along some pairs of continuous features, there was some level of linear separation between the 2 output labels.

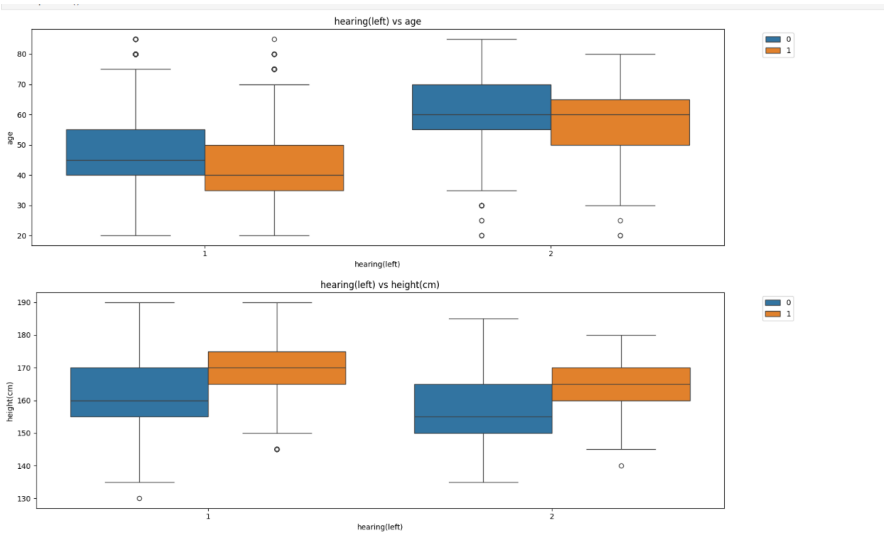


Figure 8: One Box plot among Many

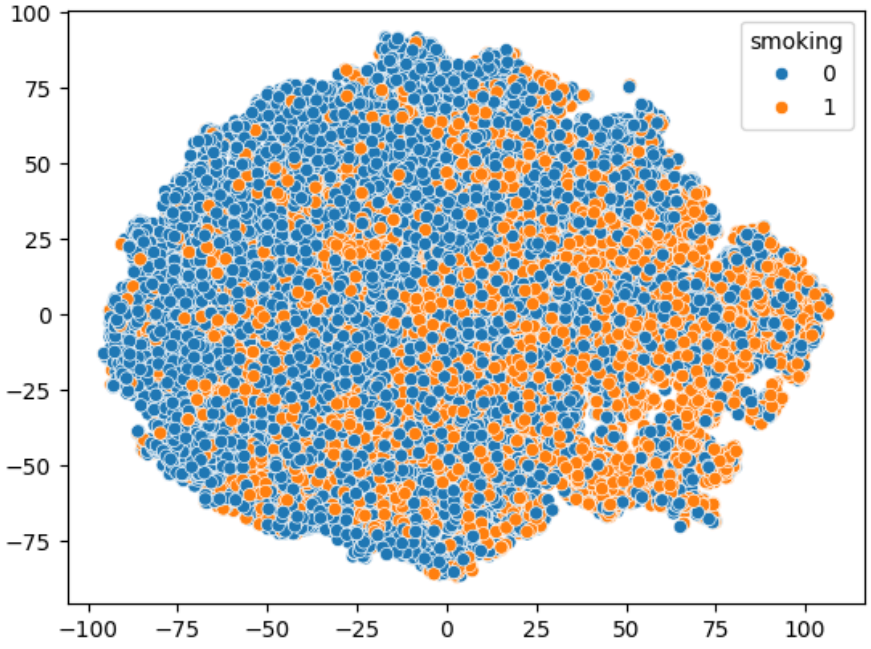


Figure 10: TSNE plot using 2 components from continuous features

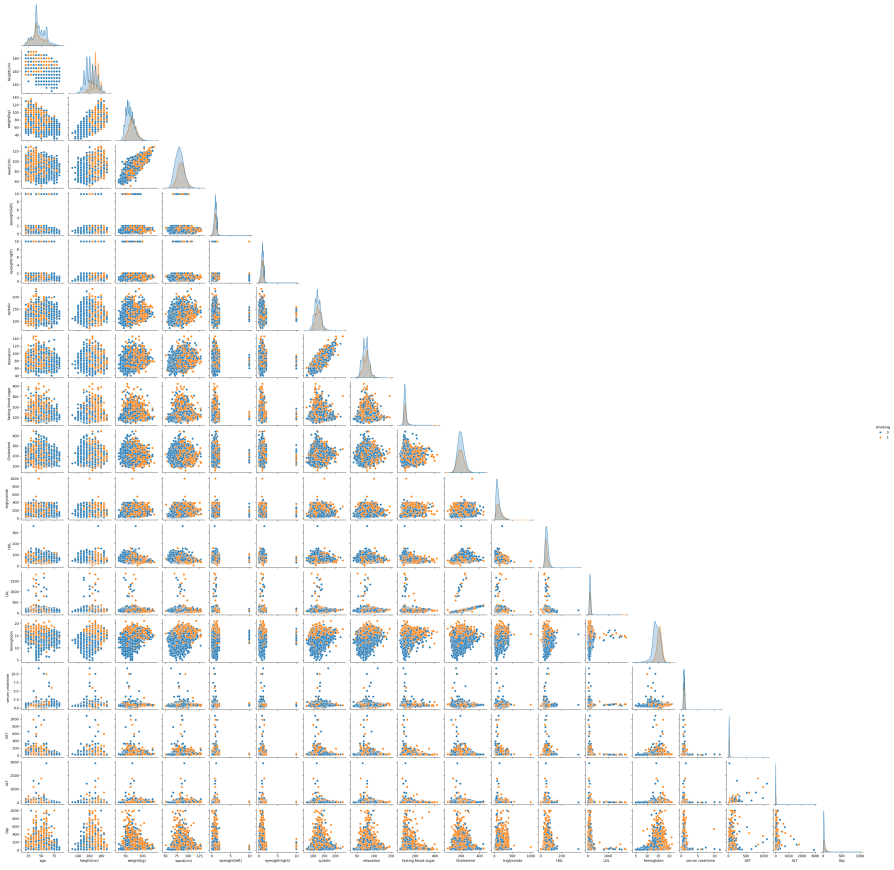


Figure 9: Pairplots of continuous features

5 Data Processing Steps

1. The dataset contained 5517 duplicate rows which were detected and dropped. There were no cells in the dataset with missing/null values.

2. The features "hearing(left)" and "hearing(right)" contained 2 distinct values- 1 and 2. This made them a categorical column, hence we converted those values to 0 and 1 respectively for transforming it into label encoding format.

3. The feature Urine protein contained integers 1 to 6, signifying different protein types. As this made it a categorical column with more than 2 values, one hot encoding was applied on it, splitting it into 6 distinct columns.

4. The output column "smoking" was already label encoded as 0 for non-smoker and 1 for smoker.

5. MinMaxScaler transforms each feature to a fixed range, typically [0, 1], by

subtracting the minimum value and dividing by the range (max - min) of the feature. This results in a normalized dataset where the smallest value becomes 0 and the largest becomes 1, preserving the original shape of the distribution. It is particularly useful when the data has a bounded range, such as image pixel intensities (0–255), or when the algorithm expects inputs within a specific interval, like neural networks. However, MinMaxScaler can be sensitive to outliers, as they compress the inlier values into a narrow range. StandardScaler, on the other hand, standardizes features by removing the mean and scaling to unit variance, resulting in a distribution with a mean of 0 and a standard deviation of 1. This is achieved by subtracting the mean and dividing by the standard deviation of each feature. It is ideal when the data follows a Gaussian (normal) distribution or when the algorithm assumes normally distributed inputs, such as logistic regression, support vector machines, or principal component analysis (PCA). StandardScaler is less sensitive to outliers compared to MinMaxScaler in some contexts, although it can still be influenced by extreme values. In practice, the choice between MinMaxScaler and StandardScaler often depends on the data distribution and the machine learning algorithm used. If the data is normally distributed, StandardScaler is typically preferred. If the data has a bounded range or is non-Gaussian, MinMaxScaler may be more appropriate. For many cases, especially with algorithms sensitive to feature scale, StandardScaler is a safe default choice. Therefore Standard scaling was performed on continuous features.

6. As the dataset was split into train and test sets within the Cross-Validation implementation of GridSearchCV, scaling before model was trained would cause data leakage between and train and test sets. Therefore, instead of directly passing the model, a pipeline was created using scikit-learn to ensure that scaler was both fit and transformed only on train set while only transformation was applied on test set features.

7. For logistic regression and support vector classifiers, we use stratified k fold split of 5 folds to split the data into train and test sets. This means every iteration, train and test sets were split as 80 and 20 percent respectively.

8. For neural network classifier, we used stratified k fold split of 3 folds instead with the validation fraction parameter set to 0.25 for early stopping. This implies every iteration, train-validation-test splits were about 50, 17 and 33 percent respectively.

9. In the "V2_FeatureSelection" notebook, we experiment by dropping the columns 'waist(cm)' and 'systolic' from the dataframe as they had more than

75% correlation with 'weight' and 'relaxation' features respectively while having lower correlation with the target feature compared to them.

10. As removing highly correlated features decreased the performance slightly rather than contributing to improvement, in the "V3_OutlierRemoval" notebook, we experiment by removing points we deemed outliers based on box plots of "hearing(left)" and "Urine protein" features in the EDA notebook. 236 points were removed.

6 Hyperparameter Tuning

1. Grid search was used for hyperparameter selection as our search space was not that large hence we could afford to do an exhaustive lookup. Stratified K Fold cross validation was used to ensure that class imbalance does not affect model performance.

2. Wherever needed, random state was fixed as the standard number of 42.

3. Hyperparameters such as learning rate and regularization term (C or alpha) tend to be very small positive numbers. When we sample them, we would like to sample values from all the orders of magnitude in a given interval. Let's say we decide to sample uniformly from 0 to 1, then only about 10% of the values would come from 0 to 0.1, and 90% of the values would come from 0.1 to 1. Instead, if we used a logarithmic scale to sample the values such as from -5 to 0, then values from 1e4, 1e3, 1e2, 1e1, 1e0 all have equal chance of being selected. A general rule of thumb derived from this is to sample values that are derived by multiplying 3 constantly to the initial smallest value which is in the form of 1e-x where x is a natural number, such as [0.01, 0.03, 0.1, 0.3] where x=2.

4. In the scikit-learn implementation of Logistic Regression, learning rate is not specified as the class uses methods to choose and adapt learning rate internally for best performance.

5. We test all types of regularization on logistic regression- None, L1, L2 and ElasticNet. The default solver 'lbgfs' works with None and L2 but the solver 'saga' is required to implement L1 and by extension Elastic net.

6. While different kernel functions were checked in the V1 notebook for support vector classifier, it was observed that Radial Basis Function (RBF) substantially and clearly outperformed other kernels. Hence to preserve time, in the V2 and V3 notebooks we fix the kernel as RBF and only tune the regularization parameter C.

7. In neural networks, it is a convention to choose numbers that are powers of 2 as number of nodes in a hidden layer. Therefore, we choose the combinations of 1-hidden-layer and 2-hidden-layer with 64 and 128 neurons each to test. We also check how the sigmoid, tanh and relu activation functions affect performance.

8. The max number of iterations for logistic regression, svm and neural networks were set as 10000, 100000 and default. This is needed to limit the training time taken.

7 Comparative Analysis

Table 1: Model Performance (V1 Notebook)

Model	**Best Mean Test Accuracy**	**Optimal Hyperparameters**

Logistic Regression	0.7211	`C = 1`
		`l1_ratio = 0.25`
		`penalty = 'l1'`
		`solver = 'saga'`

Support Vector Classifier	0.7501	`C = 1` `kernel = 'rbf'`

Neural Network	0.7504	`activation = 'tanh'`
		`alpha = 0.001`
		`hidden_layer_sizes = (64, 64)`
		`learning_rate_init = 0.001`

Table 2: Model Performance (V2 Notebook)

Model	**Best Mean Test Accuracy**	**Optimal Hyperparameters**

Logistic Regression	0.7211	`C = 1`
		`l1_ratio = 0.25`
		`penalty = 'l1'`
		`solver = 'saga'`

Support Vector Classifier	0.7481	`C = 1` `activation = 'relu'`

Neural Network	0.7417	`alpha = 0.1`
		`hidden_layer_sizes = (128,)`
		`learning_rate_init = 0.03`

Table 3: Model Performance (V3 Notebook)

Model	Best Mean Test Accuracy	Optimal Hyperparameters

Logistic Regression	0.7222	`C = 0.1`
		`l1_ratio = 0.25`
		`penalty = 'l1'`
		`solver = 'saga'`

Support Vector Classifier	0.7501	`C = 1`

Neural Network	0.7435	`activation = 'relu'`
		`alpha = 0.01`
		`hidden_layer_sizes = (128,)`
		`learning_rate_init = 0.03`

7.1 Interpretation Model-by-Model

7.1.1 Logistic Regression

V1 to V2: The accuracy remains effectively unchanged (0.7211 $\rightarrow$ 0.7211). This indicates that removing the two highly correlated features did not affect model performance. Logistic Regression, especially with an L1 penalty, handles multicollinearity well, and the removed features did not contribute additional independent predictive information.
V1 to V3: A slight improvement is observed (0.7211 $\rightarrow$ 0.7222). Removing outliers produces a cleaner linear decision boundary, which benefits a low-capacity, parametric model such as Logistic Regression.
Conclusion for LR: Dropping correlated features has no effect, while removing outliers produces a small but positive gain.

7.1.2 Support Vector Classifier (RBF Kernel)

V1 to V2: A small decrease in performance is observed (0.7501 $\rightarrow$ 0.7481). Unlike Logistic Regression, nonlinear models such as SVC can leverage correlated features as additional dimensions along which the RBF kernel can separate data. Removing features therefore slightly reduces useful structure.
V1 to V3: Performance returns to its original level (0.7501). Outlier removal helps SVC, which is sensitive to extreme values that distort the margin and the kernel's distance calculations.
Conclusion for SVC: Dropping correlated features is mildly harmful, while removing outliers restores and stabilizes performance.

7.1.3 Neural Network

V1 to V2: A notable drop in accuracy is observed ($0.7504 \rightarrow 0.7417$). Neural networks thrive on richer feature spaces, including correlated features, because they provide redundant signals and allow deeper representations. Removing features reduces model expressiveness and harms performance.
V1 to V3: A partial recovery occurs ($0.7417 \rightarrow 0.7435$), but the model does not return to its V1 performance. Although removing outliers reduces noise, neural networks are highly data-dependent. Reducing the dataset size by discarding outliers harms generalization more than it helps, unless the dataset is extremely large.
Conclusion for NN: Dropping correlated features is strongly detrimental. Outlier removal provides mild improvement over V2 but does not restore V1 accuracy, indicating that sample size reduction and information loss outweigh benefits from noise reduction.

7.2 Cross-Model Interpretation by Preprocessing Strategy

7.2.1 Dropping Highly Correlated Features (V2)

Effects:

- LR: No effect
- SVC: Slightly negative
- NN: Strongly negative

Interpretation: The dropped features were not redundant for nonlinear models. SVC and NN likely used them to capture nonlinear relations and redundant encodings useful for generalization. Only LR, which cannot exploit such relationships, remained unaffected.
Conclusion: Avoid dropping correlated features unless using strictly linear models or unless multicollinearity diagnostics (e.g., VIF) strongly justify removal.

7.2.2 Removing Outliers (V3)

Effects:

- LR: Slight improvement
- SVC: Clear benefit, restoring best performance
- NN: Mild improvement over V2 but still worse than V1

Interpretation: Outlier removal benefits LR and SVC because their decision boundaries are sensitive to extreme values. Neural networks, however, can learn to ignore outliers when data is abundant, and removing samples harms performance when data is limited.
Conclusion: Outlier removal is beneficial for LR and SVC. Neural networks prefer outlier *mitigation* (e.g., clipping or robust scaling) rather than outright removal.

7.3 Best Overall Performer

Across all notebooks and preprocessing strategies, the highest accuracy is achieved by:

Neural Network (V1): 0.7504

Although SVC performs nearly as well, the NN in V1 settings remains the best performer. The results collectively indicate that the dataset contains meaningful nonlinear structure, that correlated features are beneficial, and that removing outliers is not optimal for models requiring large data volume such as neural networks.

8 Future Work

1. Target classes did not have heavy difference in their frequencies and despite that they could be handled using Stratified Kfold training and oversampling. However, categorical features had class imbalance among their categories. Oversampling with respect to categories of categorical features rather than target categories can be explored as a way to improve performance.

2. Identifying and removing points classified as outliers/noise by DBSCAN algorithm then training and testing the models can be explored.

3. Advanced hyperparameter tuning techniques like those involving evolutionary algorithms or bayesian optimization can be utilized for better selection.

4. Performance of other machine learning techniques like tree-based classifiers and ensembling can be checked.