In21-S4-CS3111 - Introduction to Machine Learning
K.S. RANASINGHE
210518H

Lab 01 - Feature Engineering

Introduction

We have been provided with a loan default prediction data set from a finance company in the United States. The data provided contain information on previous loan applicants and whether they 'defaulted' or not. The aim is to identify patterns that indicate if a person is likely to default, which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. Features, also known as variables, include credit scores, the quantity of financing queries, address details such as zip codes and states, and collections, among other factors. The data set is a matrix of around 860,000 observations and 150 variables. The target variable is 'loan_status' as mentioned in the datasets.

```
The shape of training dataset : (517788, 145)
The shape of validating dataset : (172596, 145)
The shape of testing dataset : (172596, 144)
```

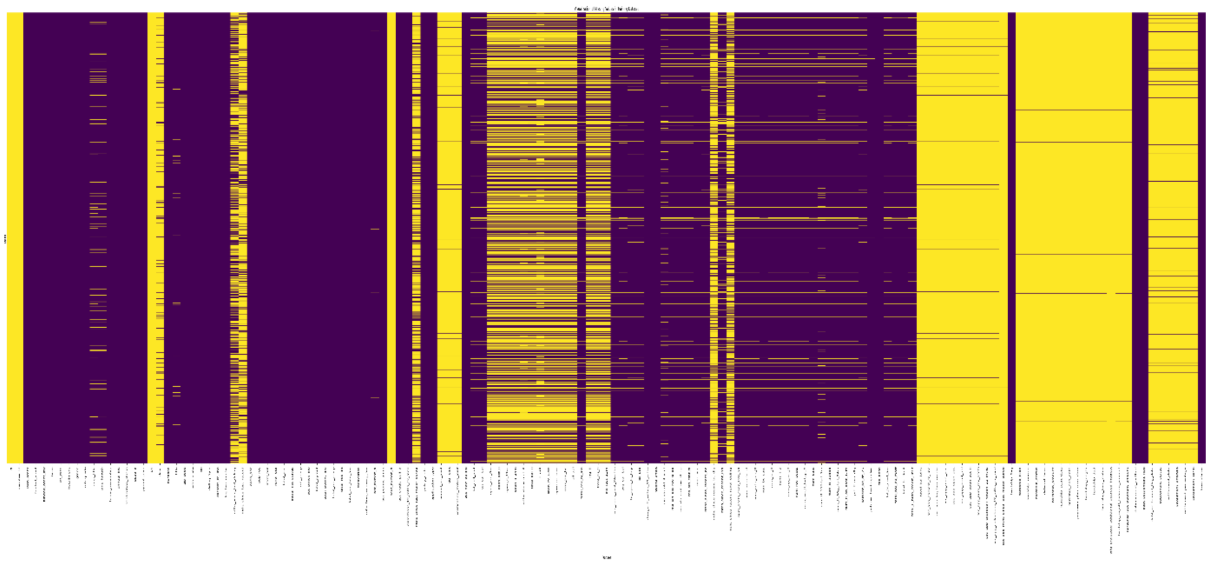
In this report, we will be working with the aforementioned datasets to build a machine learning model which will be able to predict whether a person will default on the loan they have taken based on other information collected about that person. We will be employing different feature engineering techniques to develop and fine tune our model using XGBoost Classifier and finally provide insights into the model's predictions using SHAP analysis.

Data Preprocessing

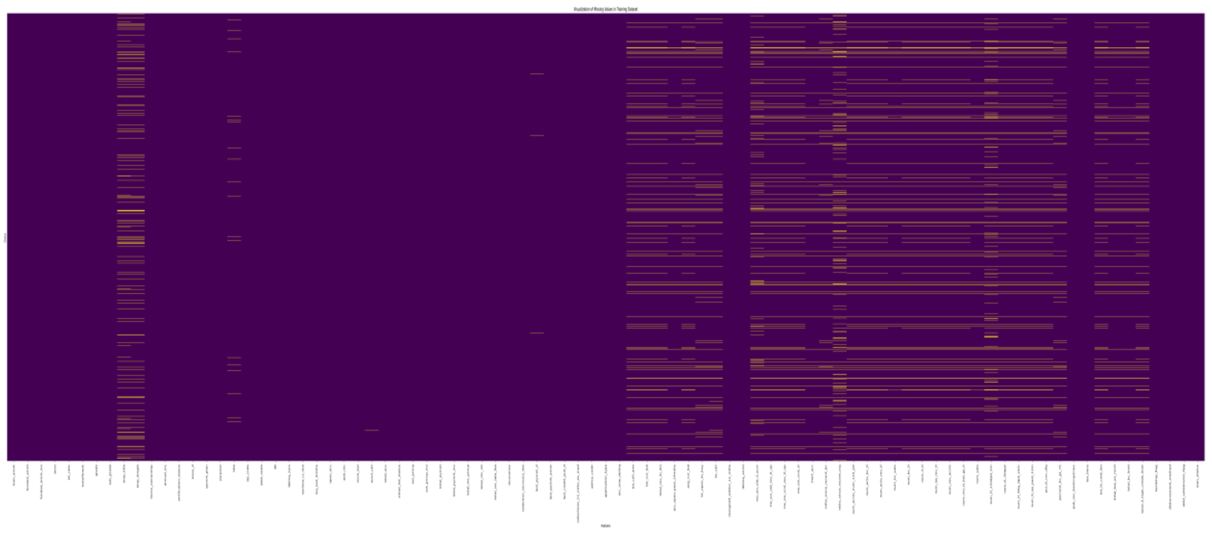
A critical phase in the machine learning process is data preprocessing, which entails getting the raw data ready for analysis and model training. It includes several steps intended to prepare the data for machine learning algorithms by organising, converting, and cleaning it. Handling missing values, encoding categorical variables, scaling numerical features, and addressing any other inconsistencies or abnormalities in the data that can impact the model's performance are the main goals of data preprocessing.
Effective data preprocessing is essential for building accurate and robust machine learning models. By ensuring that the data is clean, consistent, and properly formatted, data preprocessing helps improve the model's predictive performance, reduce overfitting, and enhance interpretability. It also enables researchers and practitioners to extract meaningful insights from the data and make informed decisions based on the model's outputs.

Handling Missing Values

Real-world datasets frequently contain missing values, which must be addressed before training a machine learning model. Missing values can introduce a bias to the predictive model as there is inadequate amount of data to make a prediction. Similarly, the loan default prediction dataset provided to us contains missing values as well. The diagram given below visualizes the dataset before handling missing values. Missing values are shown in yellow. As you can see there are many Missing values in the dataset.



There are two ways that we employed in handling missing values. As you can see above from the heatmap there are features where most if not all the data is missing. We cannot use Imputation techniques on such features as that could add a bias to the dataset since we cannot predict the performance of a feature with so little data. In our dataset we will be removing the columns with more than 50% of the data missing. Given below is the dataset after removing those columns.



Still there are missing values in the dataset. We used the Simple Imputation technique to handle the rest of the missing values. As there are both numerical and categorical features in the dataset, we had to use different imputation parameters for them. For numerical features we used the mean while we used the most frequent value for categorical features.

Feature Encoding

Feature encoding ensures the compatibility and effectiveness of categorical features in training a machine learning model. Most ML models require the input in a numerical format, therefore converting the categorical values is a requirement. Without proper encoding, categorical variables may introduce bias into the model. For example, if we encode categorical variables arbitrarily using numerical labels, the model may incorrectly learn patterns based on the magnitude of those labels, which are meaningless in the context of the original categories.

```
Number of categorical variables in the dataset: 21
```

We decided to use Ordinal Encoding technique for our dataset after closely analysing the categorical features. This technique assigns numerical values to categories based on their order or rank, preserves the ordinal relationship between categories. This is crucial for features where the order holds significance. Using techniques like One Hot Encoding on this dataset would increase the computational complexity as it would increase the number of columns.

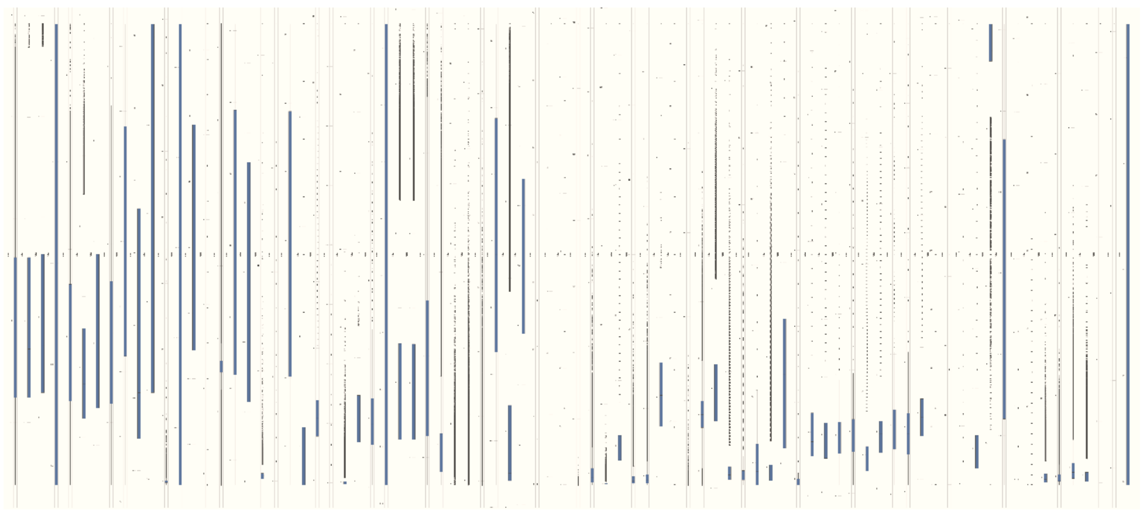
```
The shape of training dataset : (517788, 87)
```

Additionally, after converting the whole dataset to numerical values we noticed that some columns contain fixed values all throughout them. These features can be considered redundant as they have no use for the model and would be using computational resources because of their existence. So, we decided to remove them.

```
The shape of training dataset : (517788, 82)
```

Feature Scaling

Feature scaling is used to normalise or standardise the range of features or independent variables in a dataset. It is the process of converting feature values to a comparable scale. Feature scaling guarantees that the model is not sensitive to the scale of input features, helps algorithms converge more quickly during training, and keeps features with larger scales from predominating over those with smaller scales.
While both Normalization and Standardization scales the dataset very well, we decided to go ahead with Standardization as the data would be more spread out in it. Another reason for choosing Standardization is that it is a requirement for applying Dimensionality Reduction which is a part of Feature Engineering. Given below is a visualization of the dataset after Scaling.



Feature Engineering

Feature engineering is the process of adding new features to a dataset or altering pre-existing ones to enhance machine learning model performance. It entails converting unprocessed data into a format better suited for modelling, increasing the algorithm's capacity for prediction. While data preprocessing involves general data cleaning and transformation tasks, feature engineering specifically focuses on enhancing the quality and relevance of features to improve model performance.

Feature Selection

Feature selection is the process of selecting a subset of relevant features from the original set of features in a dataset. The goal of feature selection is to improve model performance by reducing the dimensionality of the data and removing irrelevant or redundant features that may negatively impact the model's ability to generalize to new, unseen data.

Feature Selection can be categorized into 3 types as Filter Method, Wrapper Method and Embedded method. I applied both Filter Method and Wrapper Method to the given dataset through SelectKBest and RFE algorithms and decided to go ahead with RFE as it performed better than the other. XGBoost Classifier was set as the estimator for the RFE algorithm.

RFE(Wrapper Method)

```
Number of features selected: 10
Selected features: Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv',
'term', 'installment',
       'total_rec_prncp', 'total_rec_late_fee', 'recoveries',
       'last_pymnt_amnt', 'debt_settlement_flag'],
      dtype='object')
Performance score: 0.9996
```

SelectKBest(Filter Method)

```
Number of features selected: 10
Selected features: Index(['int_rate', 'grade', 'sub_grade', 'total_pymnt',
'total_pymnt_inv',
       'total_rec_prncp', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_amnt', 'debt_settlement_flag'],
      dtype='object')
Performance score: 0.9947
```

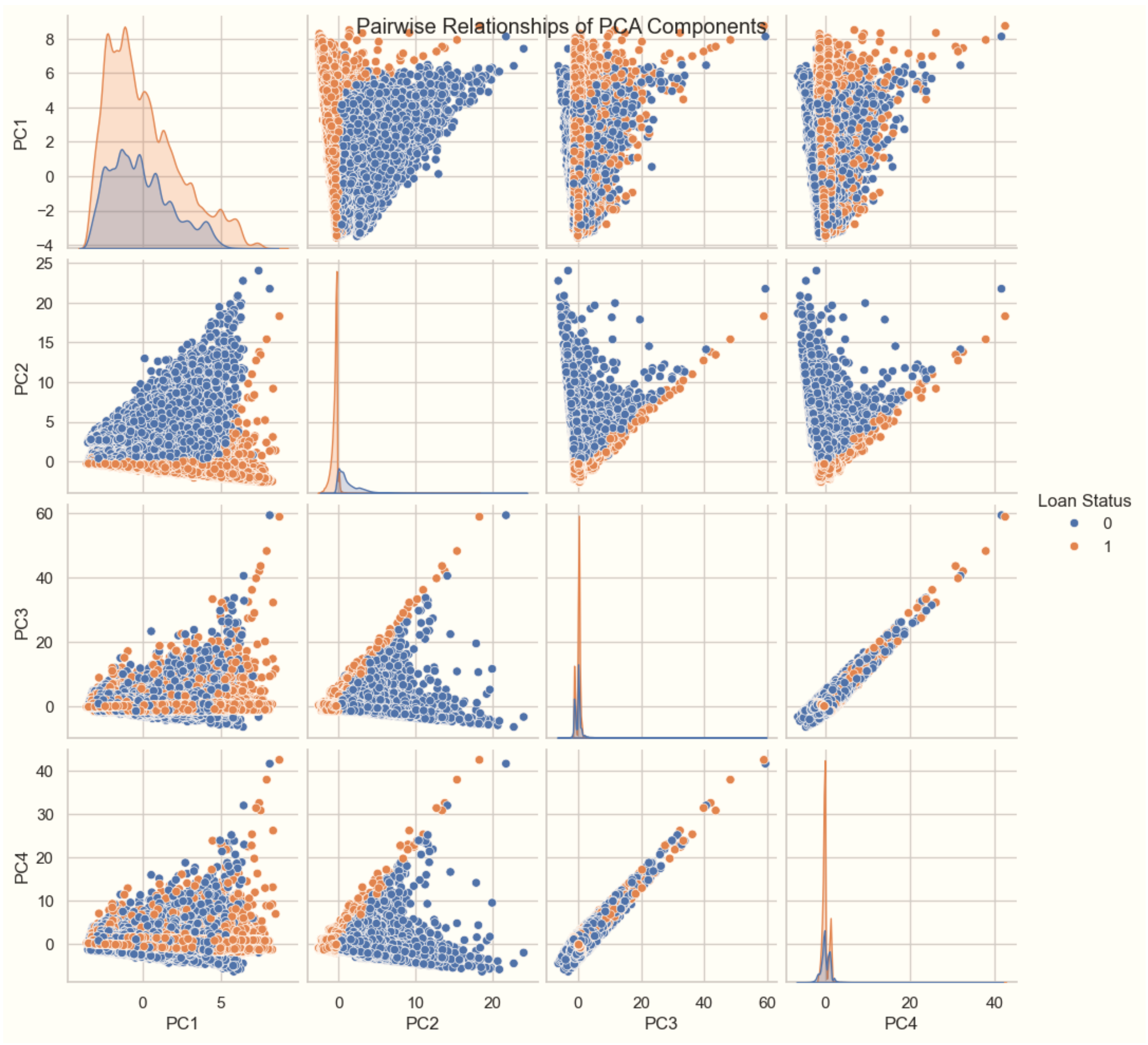
Another decision that needs to be made here was how many features to be selected. I ran the RFE algorithm for feature sizes (5,10,15,20,25) and noticed that the performance was increasing as the feature size is coming down. But having too little amount of features can prompt the ML model to overshoot it’s predictions. Therefore, I decided to go ahead with 10 features.

Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features (or dimensions) in a dataset while preserving the most important information. It is primarily used to address the curse of dimensionality, which refers to the challenges associated with high-dimensional data, such as increased computational complexity, overfitting, and difficulty in visualization. Dimensionality reduction techniques aim to capture the underlying structure or patterns in the data by transforming the original high-dimensional space into a lower-dimensional space. This transformation can be linear or nonlinear, depending on the method used.

The method I chose was Principal Component Analysis. In PCA, the algorithm identifies orthogonal axes (principal components) along which the data varies the most and projects the data onto these axes, reducing the dimensionality while preserving the maximum variance. The number of features were reduced to 4 after this process.

```
The shape of training dataset: (517788, 4)
```



XGBoost Classifier

XGBoost stands out as one of the most powerful and widely used gradient boosting algorithms in the machine learning community. Its popularity stems from its exceptional performance, scalability, and ability to handle a variety of data types. The XGBoost algorithm works by sequentially adding weak learners (decision trees by default) to a model, with each subsequent learner correcting the errors made by the previous ones. This iterative process allows XGBoost to continuously improve its predictive performance.
Using GridSearchCV, we were able to optimise the performance of our XGBoost model through hyperparameter tuning. GridSearchCV is a hyperparameter optimisation strategy that finds the set of hyperparameters that produces the greatest model performance by thoroughly searching over a given parameter grid. We sought to determine the configuration that maximises our XGBoost model's accuracy on the validation dataset by adjusting hyperparameters like max_depth, learning_rate, n_estimators, and gamma.

```
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 200, 300],
    'gamma': [0, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}
```

Given above are the set of hyperparameter values which were used for tuning the model. The final set of Hyperparameters are given below.

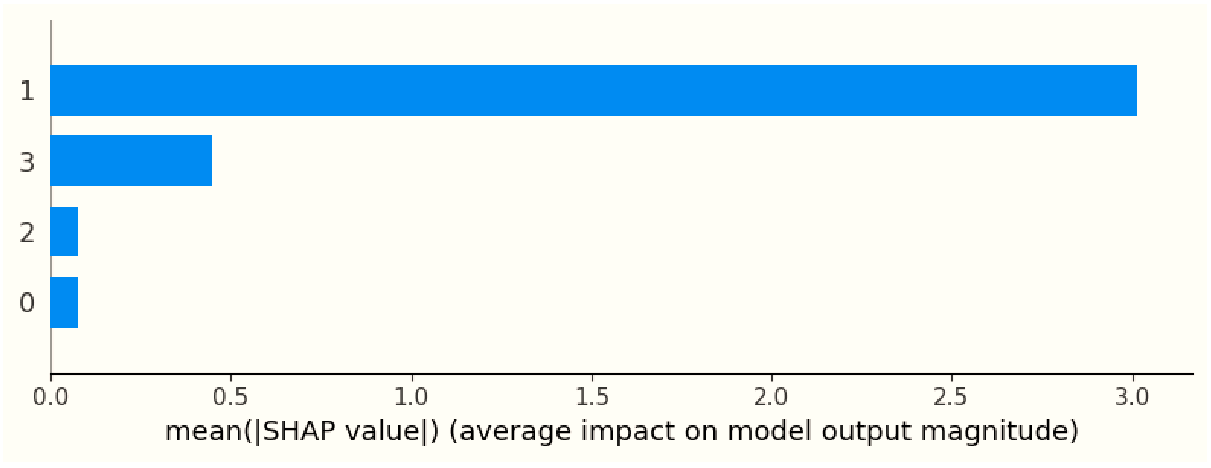
```
Best Hyperparameters: {'colsample_bytree': 1.0, 'gamma': 0.1, 'learning_rate':
0.01, 'max_depth': 7, 'n_estimators': 300, 'subsample': 0.8}
Validation Accuracy: 0.9949013882129366
```

As you can see above, the model achieved an accuracy of 99.49% on the validation dataset. Finally, we used this model to predict the target label for the Test dataset provided and wrote it into a new file as 210518H.csv. Given below is a distribution of the target label(1's and 0's) across training and testing datasets.

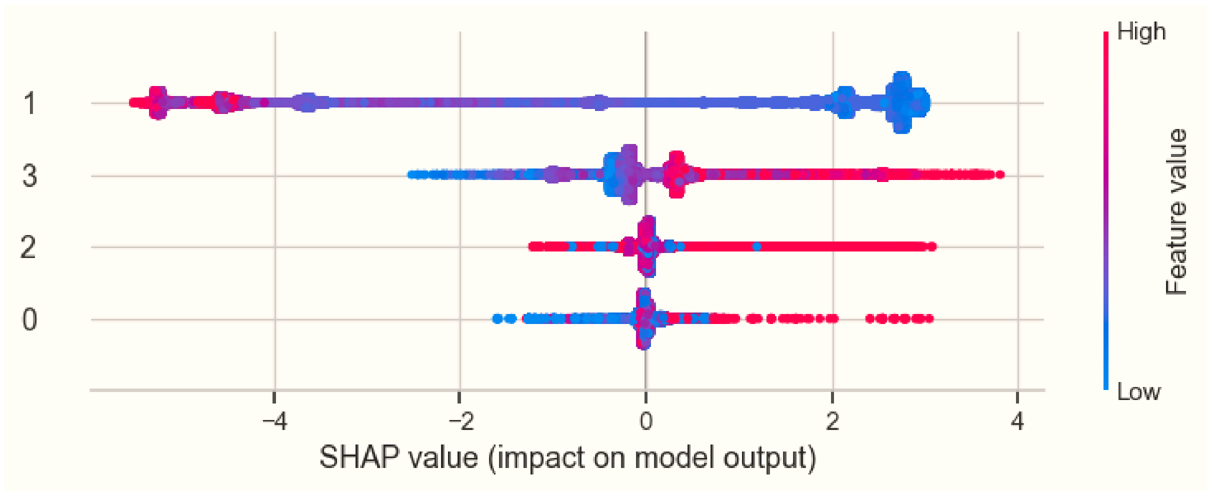
```
Predicted Label Percentages:
loan_status
1    69.60648
0    30.39352
Name: count, dtype: float64

Training Label Percentages:
loan_status
1    69.680062
0    30.319938
Name: count, dtype: float64
```

SHAP analysis for explainable AI

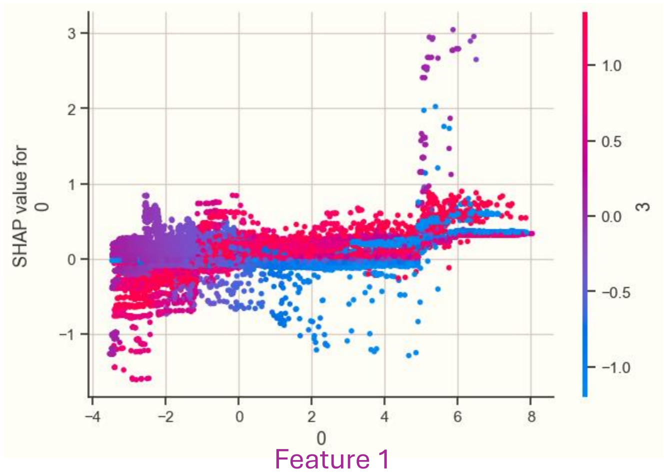
SHAP is a popular technique for model interpretability. It provides insights into individual feature contributions to model predictions. We used the SHAP library to explain the XGBoost model's predictions by visualizing feature importances, summary plots, and individual prediction explanations.



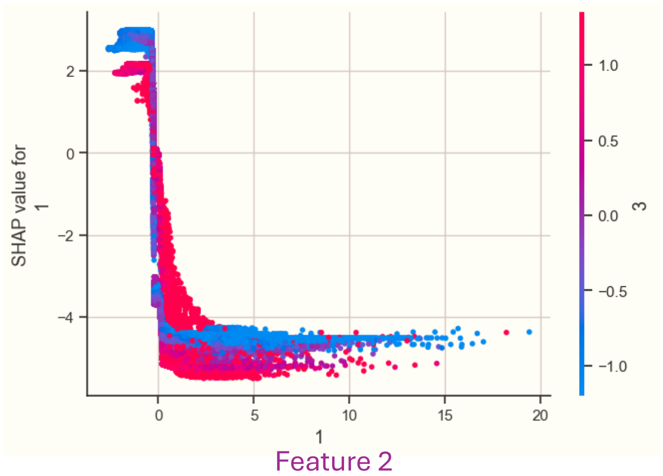
The summary plot given above provides insights into the impact of different features on the model's output across all data points. Each bar represents the mean absolute SHAP value for a feature, indicating its overall importance in making predictions.



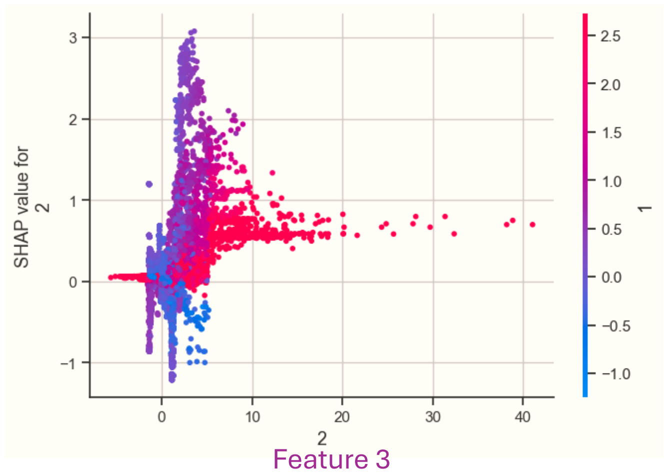
This plot provides an overview of the impact of each feature on the model's output across all data points. Each point represents a feature's SHAP value for a single prediction. The horizontal position of the point shows the magnitude of the SHAP value. The colour of the point indicates the feature value(red represents high values, blue represents low values).



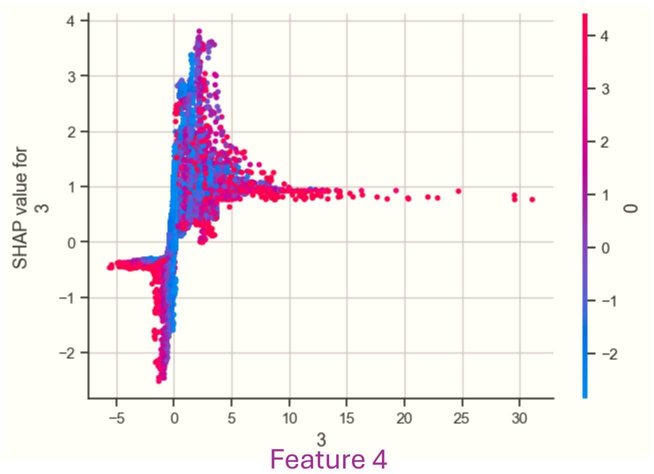
Feature 1



Feature 2



Feature 3



Feature 4

The plots given above visualizes the relationship between individual features and the model's output. It helps to understand how the model's prediction changes as the value of a specific feature varies.

Conclusion

In this project, we employed various data pre-processing techniques, feature engineering methods, and model interpretability techniques to develop a predictive model and gain insights into its predictions.

We embarked on this journey by pre-processing the raw data, which involved handling missing values, encoding categorical variables, and scaling numerical features to ensure uniformity and compatibility with machine learning algorithms.

With our pre-processed and engineered dataset in hand, we then trained a predictive model using the powerful XGBoost algorithm. Leveraging its gradient boosting framework, we fine-tuned hyperparameters using GridSearchCV to optimize model performance.

We also delved into model interpretability using techniques like SHAP, which provided valuable insights into how our model arrived at its predictions. By visualizing SHAP values, we gained a deeper understanding of the impact of each feature on the model's output, empowering us to trust and interpret its decisions with confidence.