**Problem Statement**: Aim to highlight the fraud and non fraud case. Create a predictive ML model to predict/classify if a specific vehicle insurance claim is frad or not based on historical claim details

Based on this model and EDA need to have answers to below question

● How can we analyse historical claim data to detect patterns that indicate fraudulent claims?

● Which features are most predictive of fraudulent behaviour?

● Can we predict the likelihood of fraud for an incoming claim, based on past data?

● What insights can be drawn from the model that can help in improving the fraud detection process?

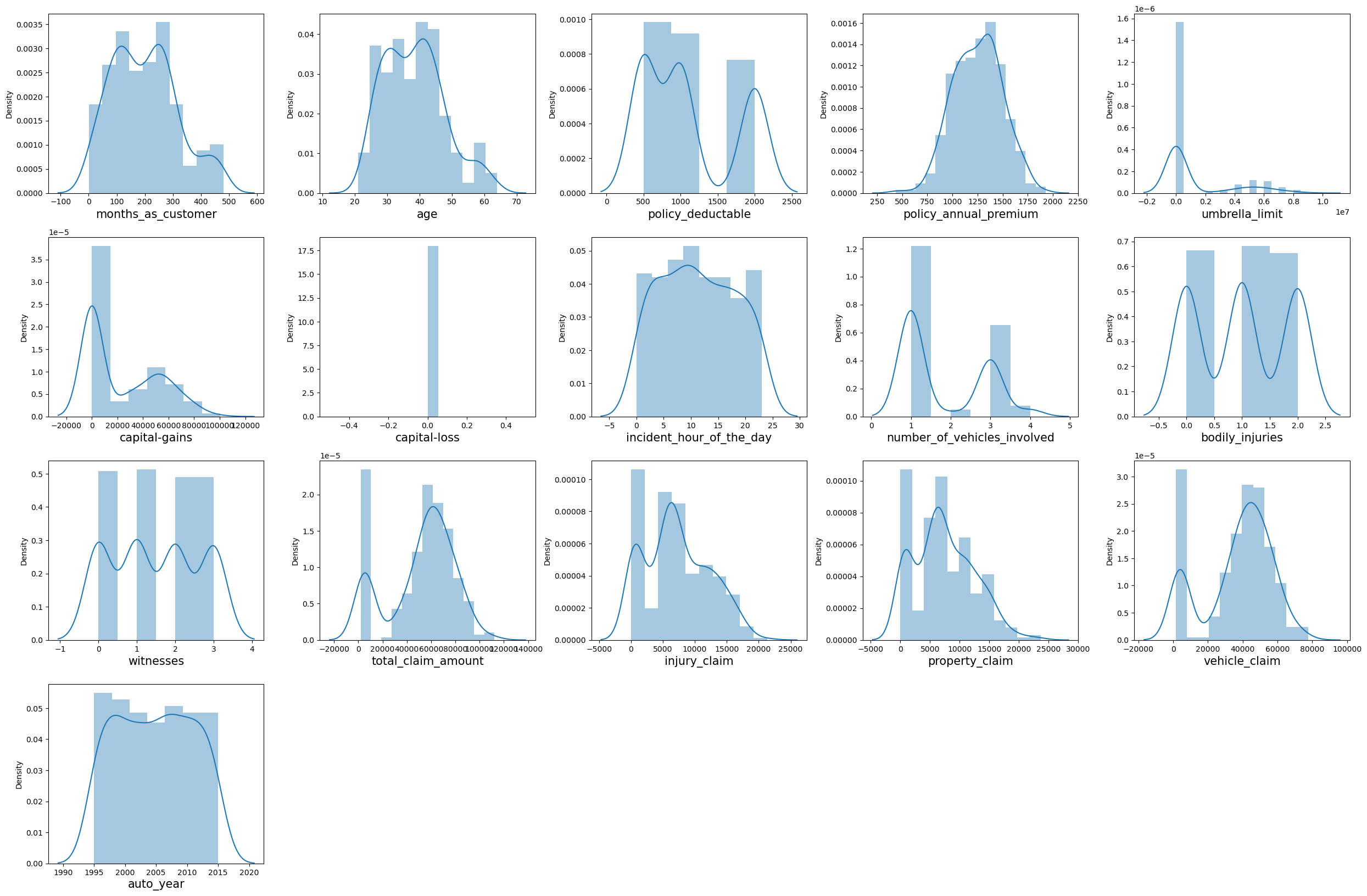
● Identify patterns , key indicators and which features total claim amount, claim type, customer profile (Age, gender, marital status etc) etc are key feature for modelling to predict fraud claim or not

**Methodology:**

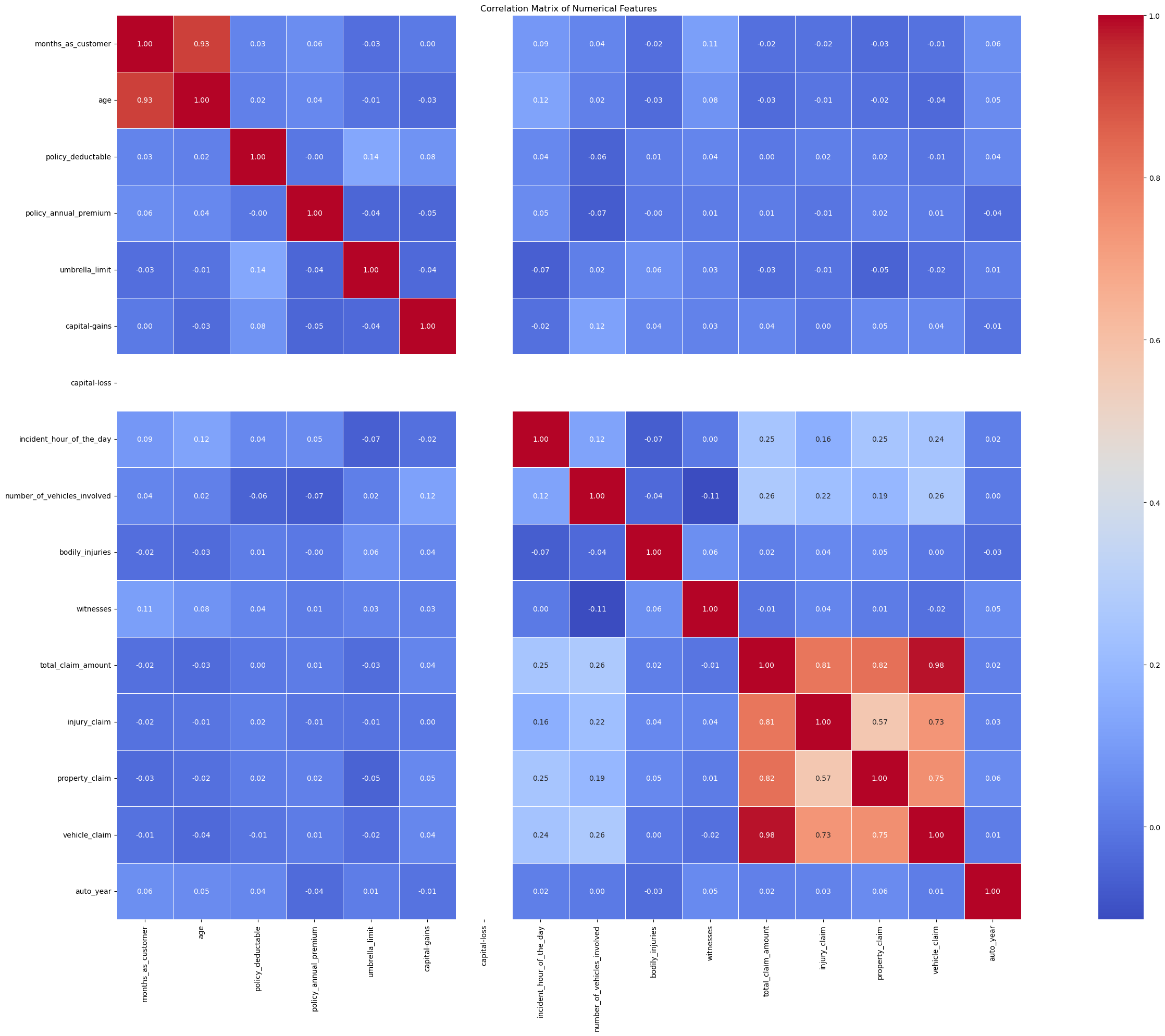
1. Data ingestion : load the data and create Data frame and understand the basic structure of data interms of shape(rows/columns), data types, which are numerical/catagorical columns. identify which columns might not be relevant for analysis if all unique values
2. Clean up and pre-process the Data
   1. handle/clean the rows containing nulls
   2. handle/clean duplicate data/rows
   3. Handle/Clean column which are all empty or majorly all unique values in column which might not be able to contribute to the model
   4. check for invalid data
      * negative values (which does not make sense to be negative)
      * Having invlaid values (? etc)
   5. fix the data types for datetime, int , bool etc
   6. NOTE: Ideally should do the EDA first and then train and test split but as per the code template we'll do test split first
3. Do Train-Test split
   1. Define feature and target variable
   2. Accordingly split the data in training and test/validation data as per independant features (X) and dependent/target feature (Y)
4. Do EDA on the data to understand the data well
   1. Univariate Analysis (single variable) : Check for all numerical features/columns and plot them to understand their characteristics
   2. Do correlation analysis to get relationships between numerical features to identify potential multicollinearity or dependencies. Visualise the correlation structure to gain insights into feature relationships.
   3. Check class balance distribution of the target variable values
   4. Bivariate Analysis
      * calculate and analyse the target variable likelihood for categorical features and help identify which are not having much variations and is uninformative
      * Similarly do for numerical features i.e how fraud\_reported (Y/N) is related to the numerical features
5. Now once data is cleaned and relevant columns and features (both numerical and categorical) are identified we'll do feature engineering on the selected feature
   1. Resampling: To handle class imbalance, to balance the number of samples in the minority class by randomly duplicating them, creating synthetic data points
   2. New feature creation : Create new more logical /relevant features from existing features by combining existing features for better analysis e.g from data/time ; total claims per policy etc
   3. Handle redundant columns both numerical and categorical features (can also be done data cleaning phase)
   4. Combine categories that occur infrequently or exhibit similar behavior to reduce sparsity and improve model generalisation
   5. Dummy variable creation : Transform categorical variables into numerical representations using dummy variables
   6. Feature scaling : Scale numerical features to a common range to prevent features with larger values from dominating the model
6. Model Building : Work on building 2 models Logistic and Random Forest
   1. Logistic Regression:
      * Feature Selection using RFECV – Identify the most relevant features using Recursive Feature Elimination with Cross-Validation and Fit RFECV on training data and only retain these selected features in training data set
      * Now using this selected features fit the Logistic model with the constant and view the summary Keeping an eye on p-values (values < 0.05 are usually considered significant)
      * Get the VIF (Variance Inflation Factor).High VIF values (typically > 5 or > 10) signal that a feature is highly correlated with others, which can mess with your model's stability and interpretation.
      * Once relevant feature are selected and high VIF value features are removed we can use this Logistic Regression model to predict data on trained data
      * Now using this predicted and actual value evaluate the model performance
      * accuracy
      * confusion matrix i.e how many true positive, true negatives
      * Precision , Recall
      * sensitivity, specificity
      * F1-score
      * Create ROC curve to get AUC value and try to get the ideal cuttoff for various fraud predication probabilites
      * For this Optimal cuttoff get the final predication and fianlly get teh various values for accuracy, confusion matrix, Precision, sensitivity/specificity
   2. Random Forest:
      * Create a Base Random Forest
      * From this Base random forest get the important features , based on some importance threshold
      * Train the model with these selected features
      * Generate the prediction on the training data and finally evaluate the model performance
      * accuracy
      * confusion matrix i.e how many true positive, true negatives
      * Precision , Recall
      * sensitivity, specificity
      * F1-score
      * Check if the model is overfitting training data using cross validation. We'll use K-Fold Cross Validation (e.g., 5 folds) on the model trained with selected features and compare the training vs cross-validation scores. If training accuracy is much higher than mean cross-validation accuracy, it could indicate overfitting. Ideally, both scores should be close — meaning the model generalizes well.
      * Enhance the performance of the random forest model by systematically exploring and selecting optimal hyperparameter values using grid search
      * Finally regenerate random forest again with these and
7. Model Prediction on Test data and calculate performance parameters

**Key Insights & Snapshots**:

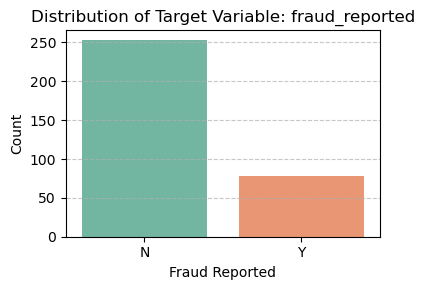
* 1. **EDA:**
     + **Univariate Analysis**

****

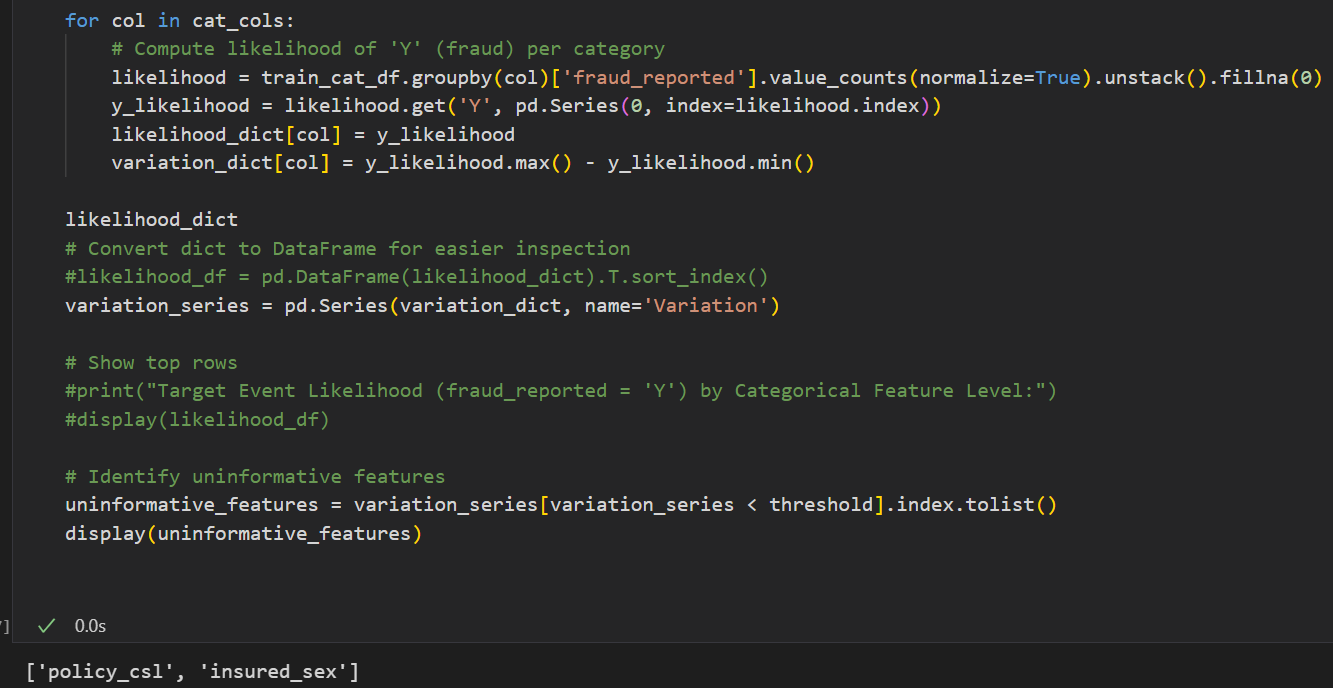
* + - **Correlation Analysis**

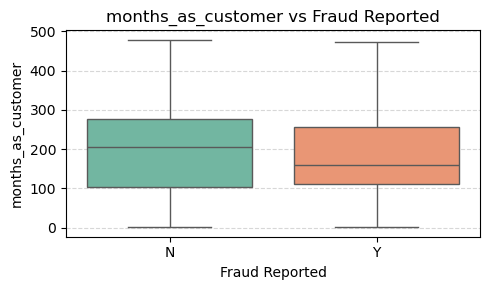
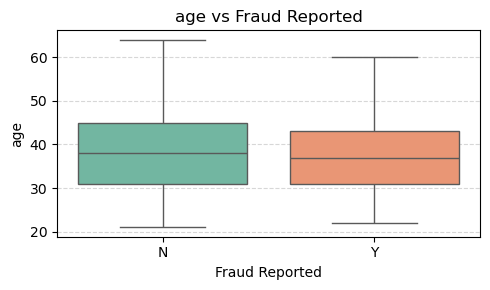
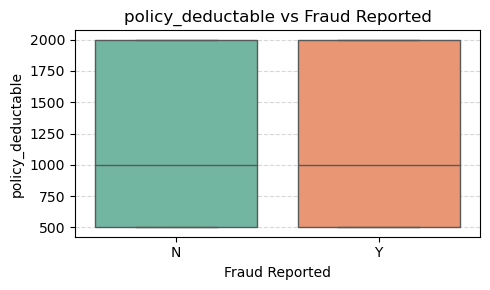
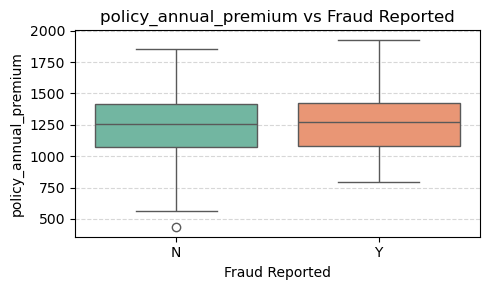
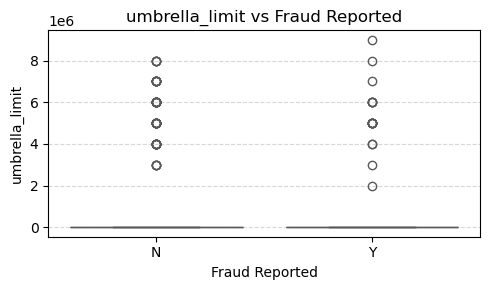
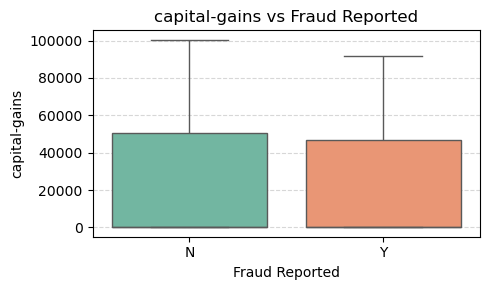
****

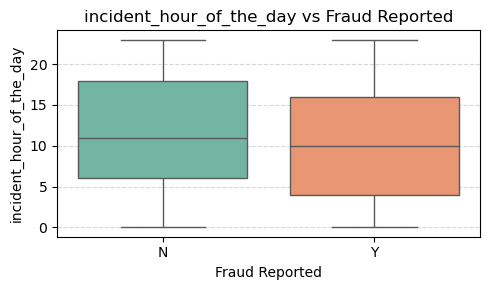
* + - **Class Imbalance**

****

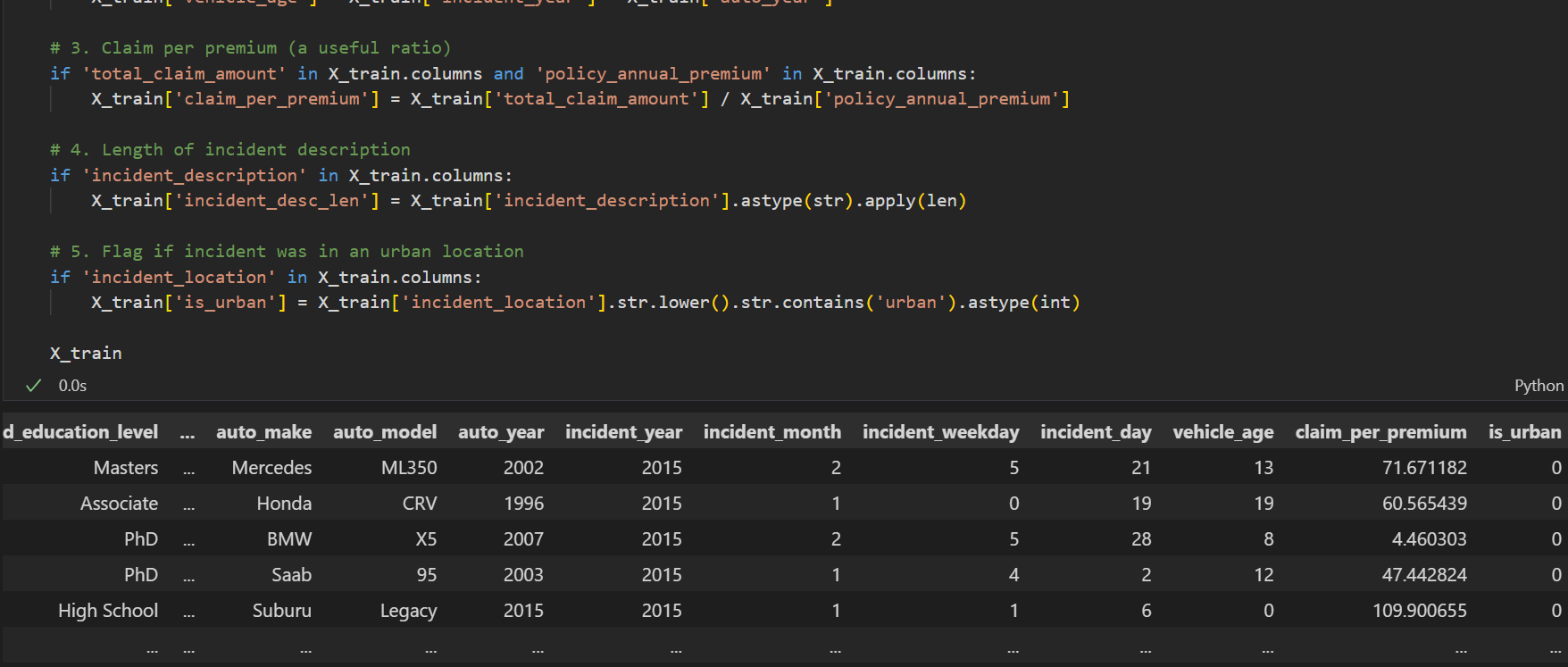
* + - **Bivariate Analysis**

****

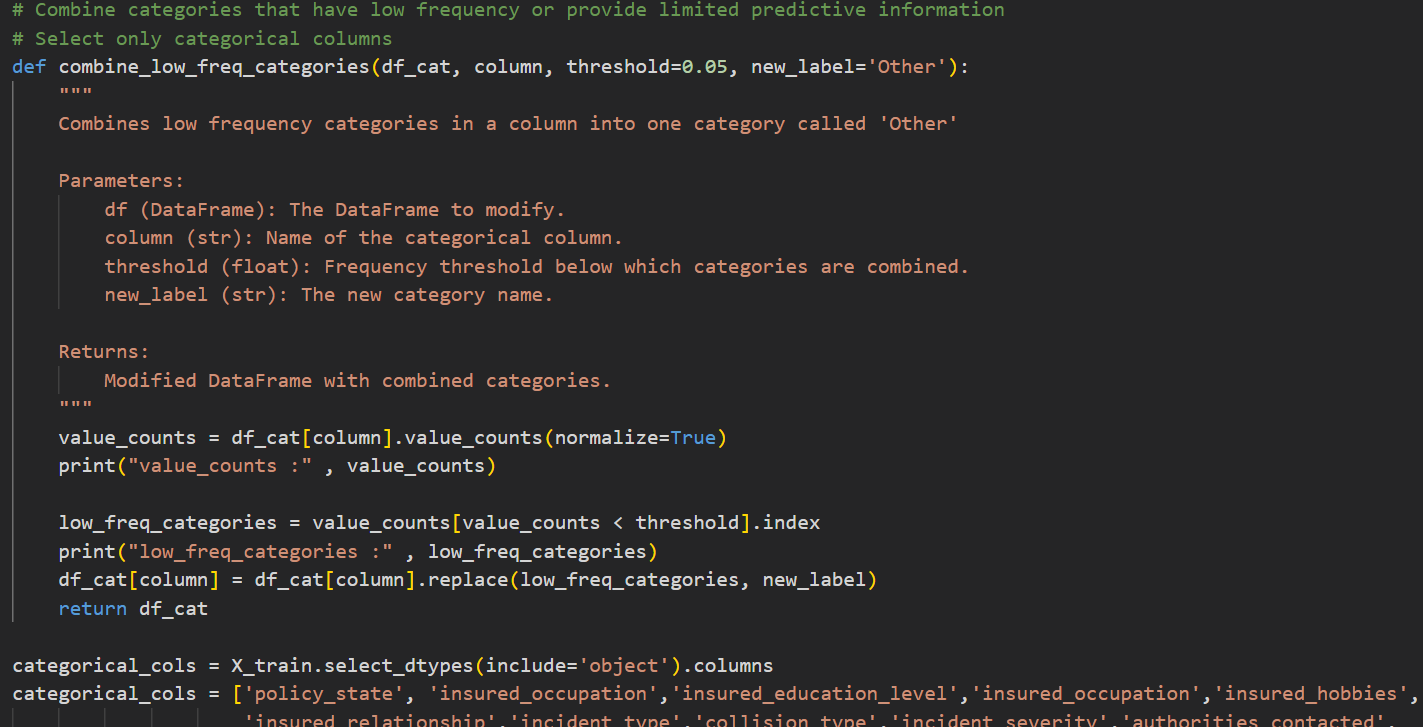
** ** ****   

****

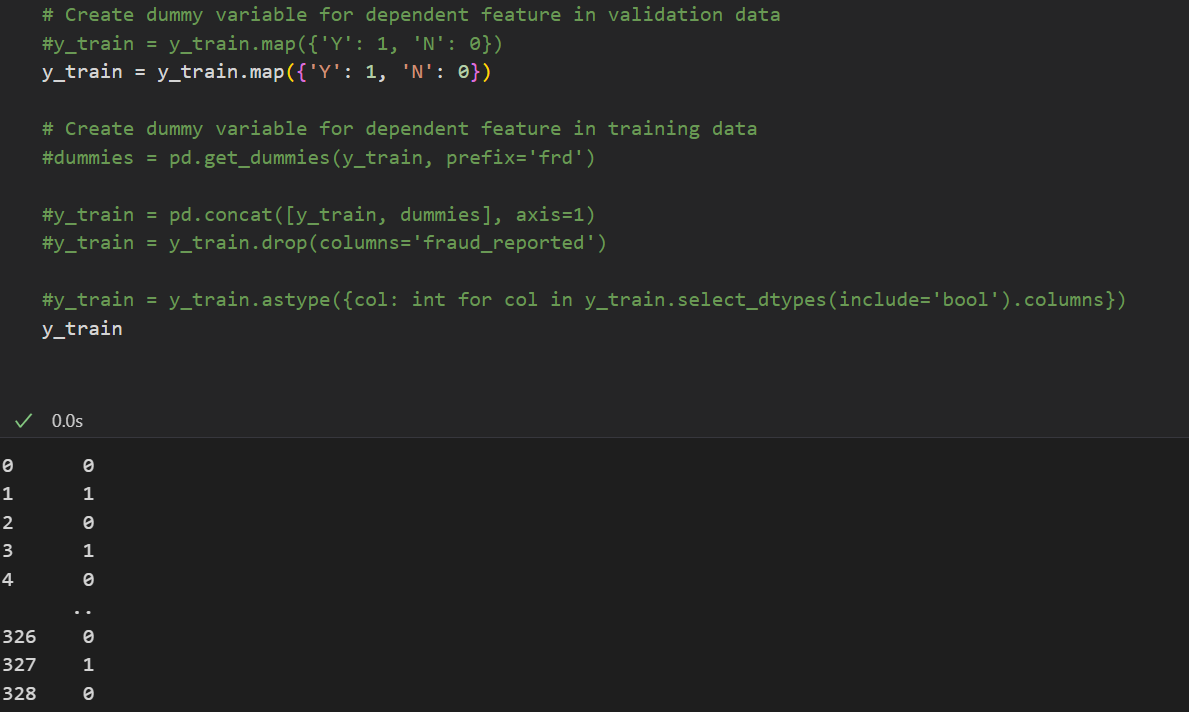
* 1. **Feature Engineering:**
     + **Resampling**
     + **New Feature Creation**

****

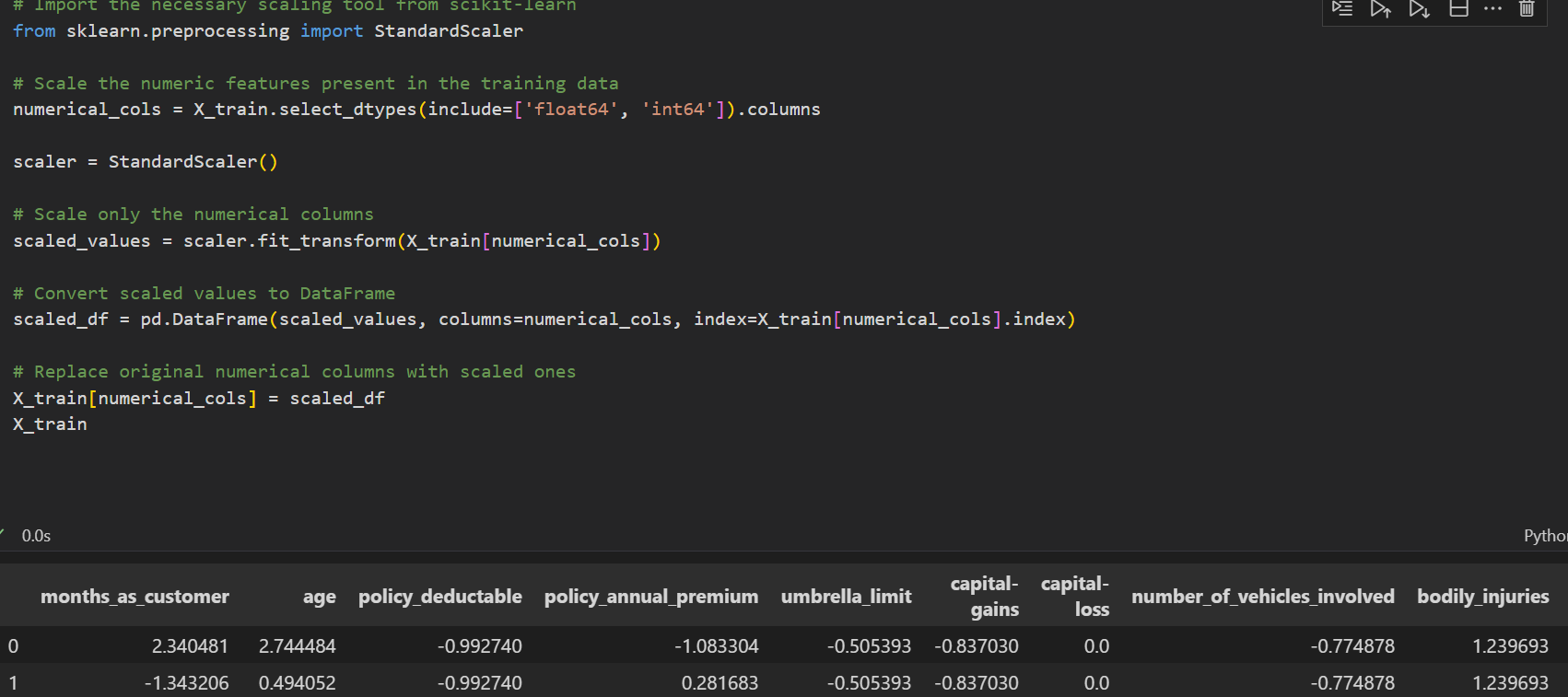
* + - **Combine Columns**

****

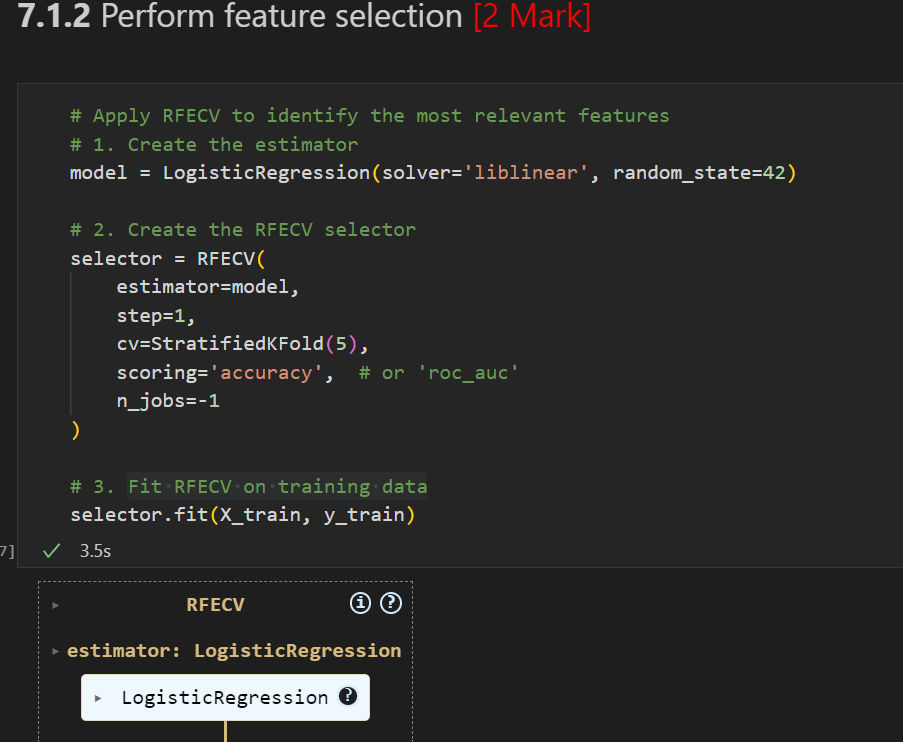
* + - **Dummy variable creation**

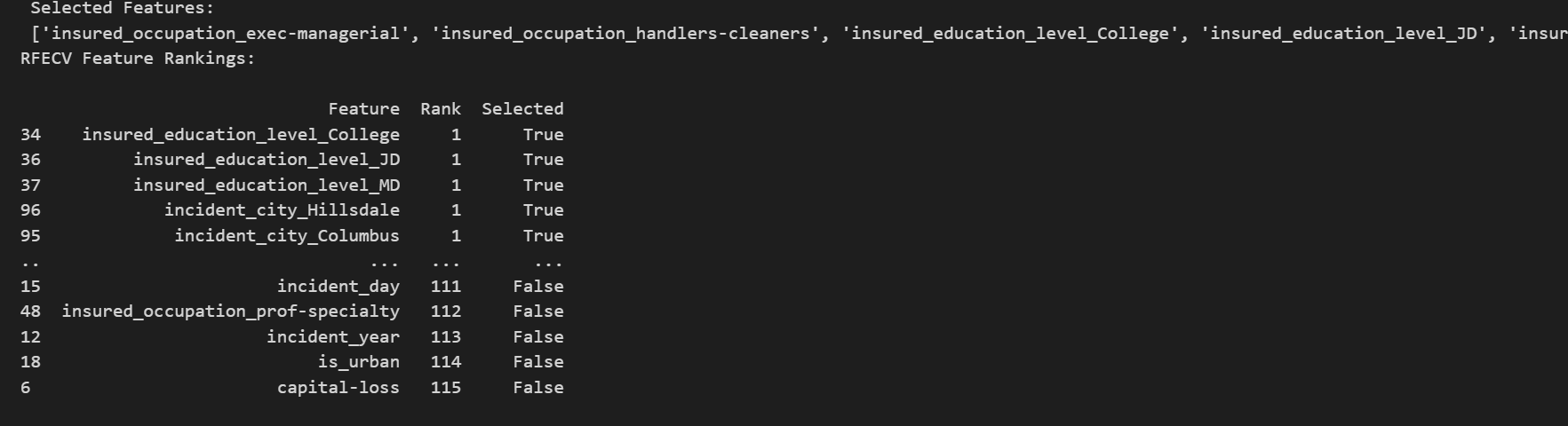
****

* + - **Feature Scaling**

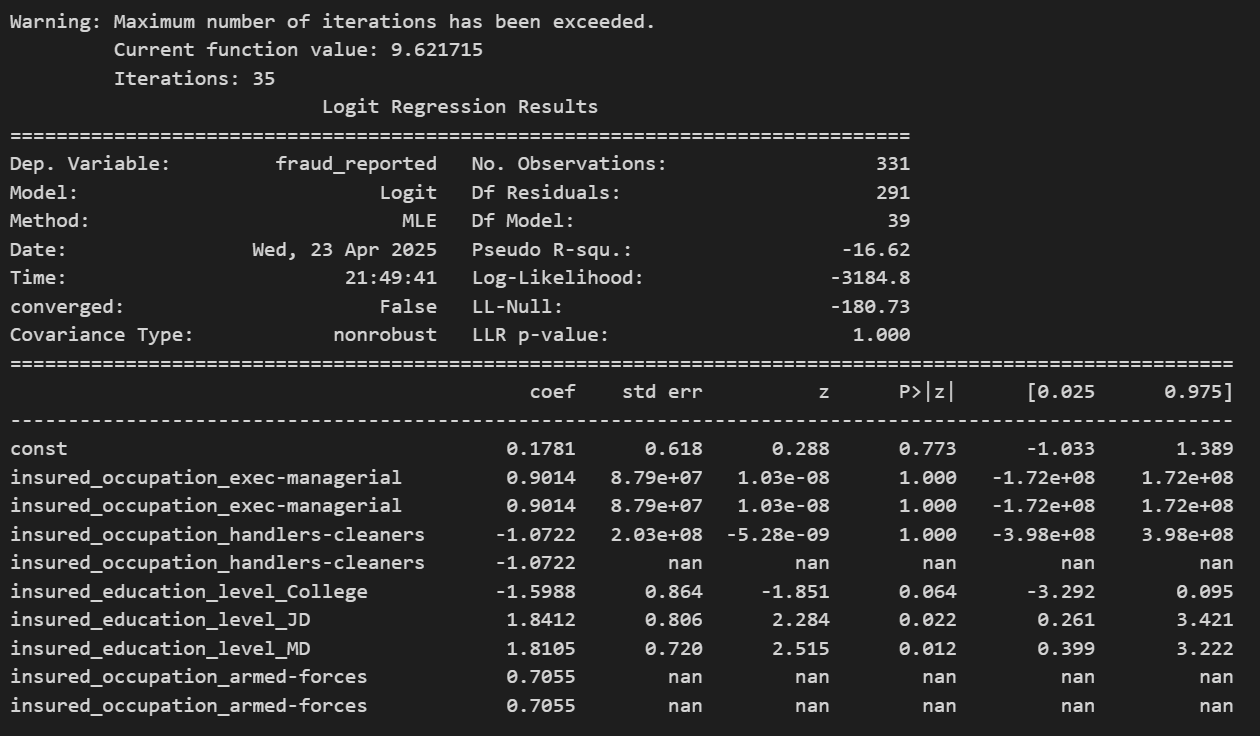
****

* 1. **Model Logistic Regression**
     + **Selected features**

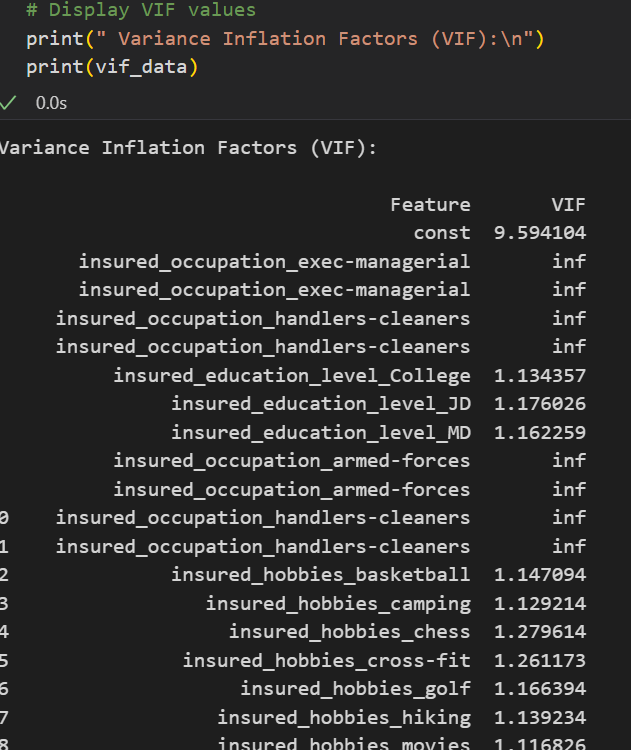
****

****

* + - **Initial Model Summary**

****

* Following details are shown in summary
  + Coefficients for each selected feature
  + p-values (to test significance)
  + Odds ratios
  + Confidence intervals
  + Keep an eye on p-values (values < 0.05 are usually considered significant)
    - **VIF**

****

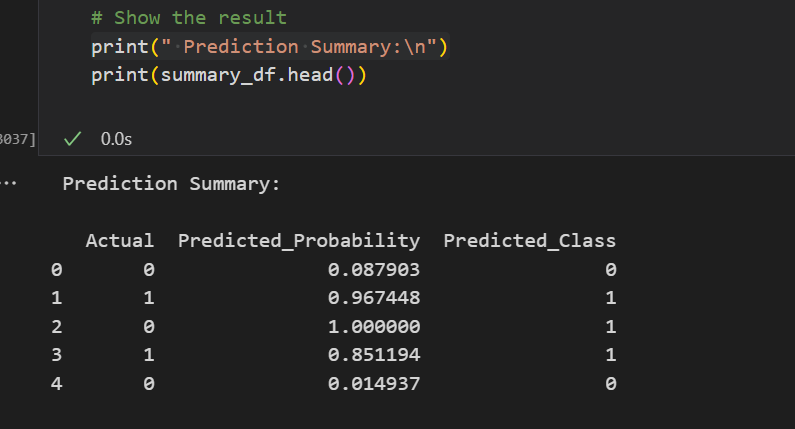
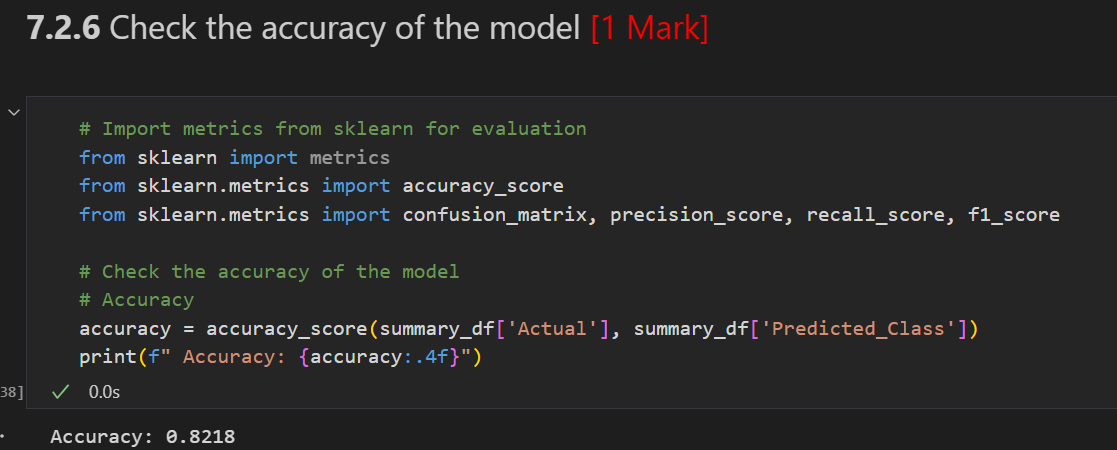
**Feature Interpretation**

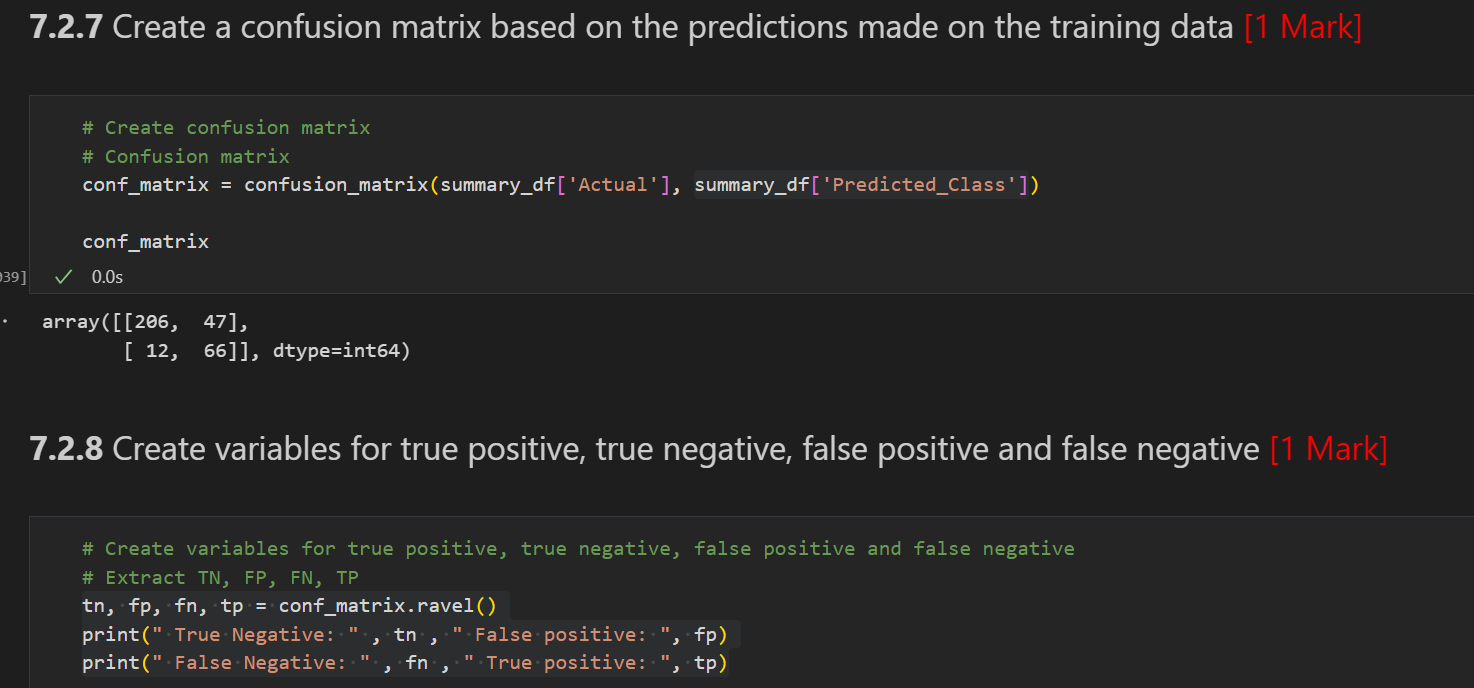
≈ 1 No multicollinearity

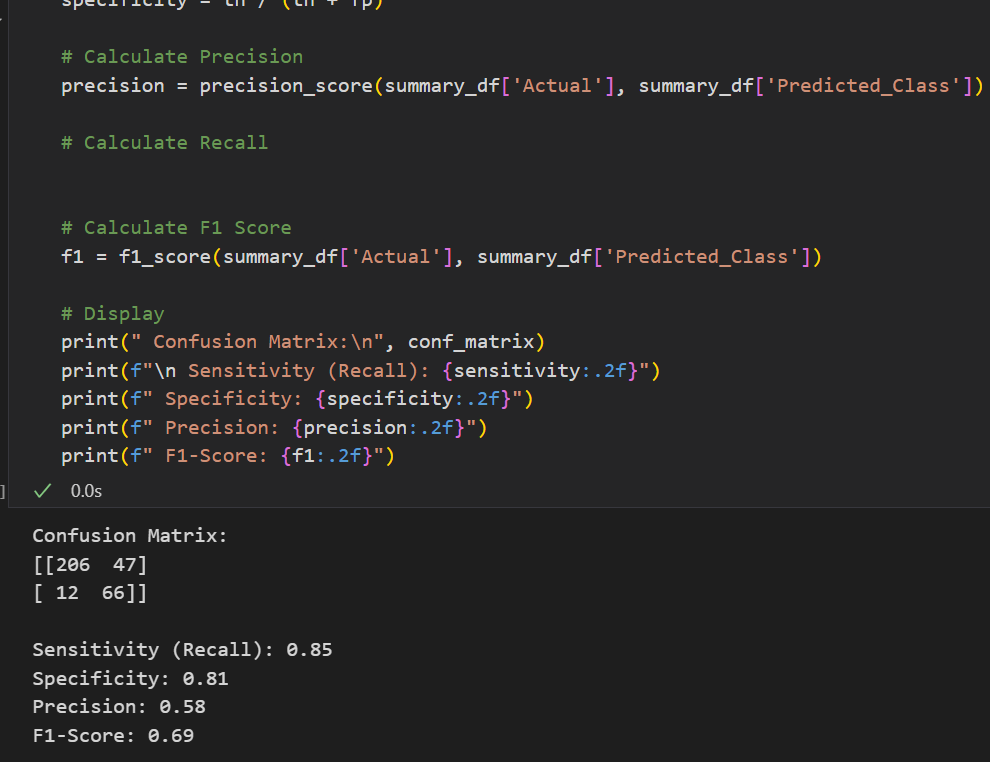
1 – 5 Moderate, usually okay

> 5 (or > 10) High, may consider removing

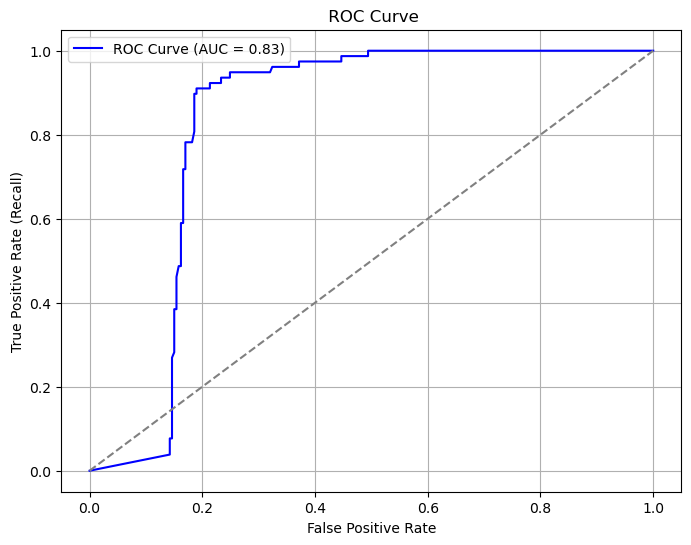
* + - **Model performance parameters**

** **

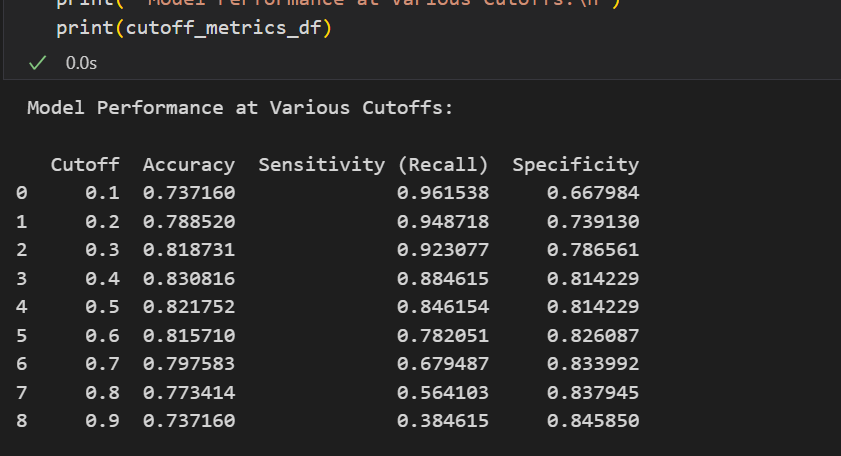
****

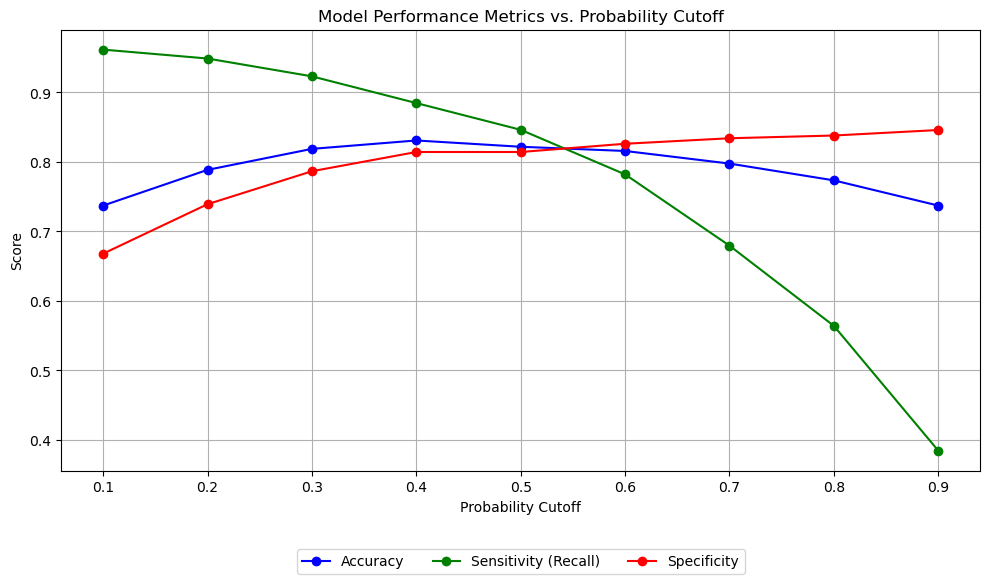
****

* + - **ROC curve**

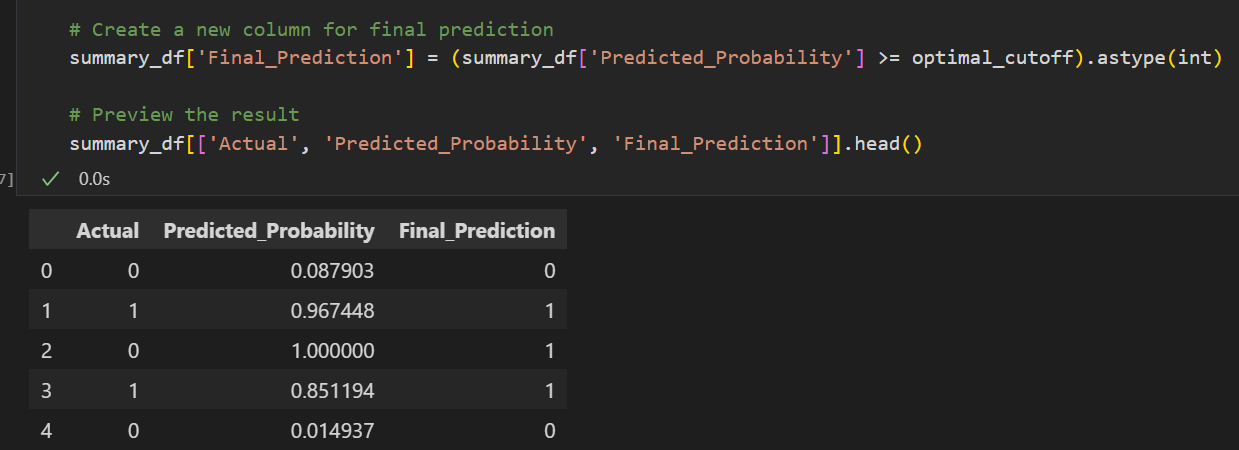
****

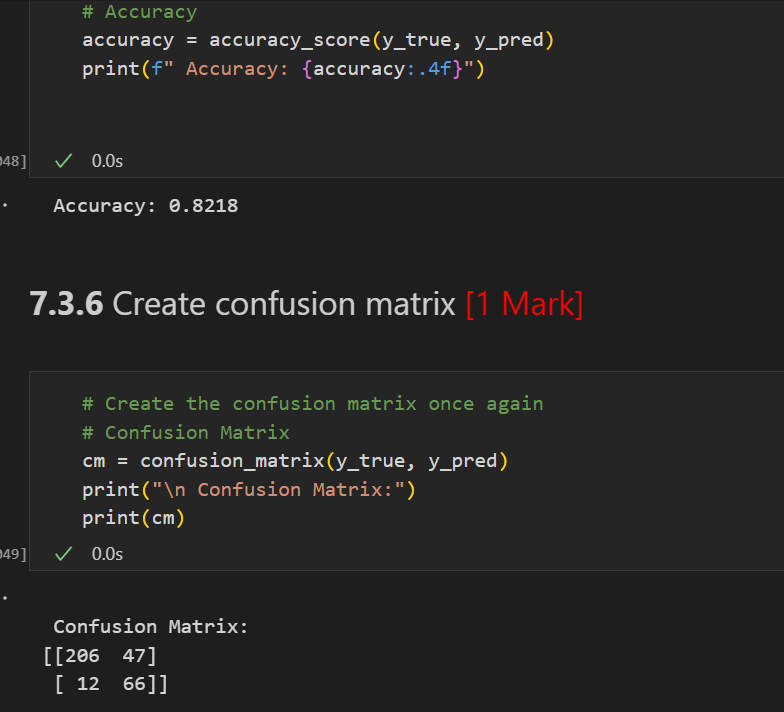
* The closer the ROC curve is to the top-left, the better the model.
* AUC (Area Under the Curve) ranges from 0.5 (random guess) to 1.0 (perfect model).
* 0.90+ → Excellent
* 0.80–0.90 → Good
* 0.70–0.80 → Fair
* 0.50–0.70 → Needs improvement
  + - **Optimal Cutoff**

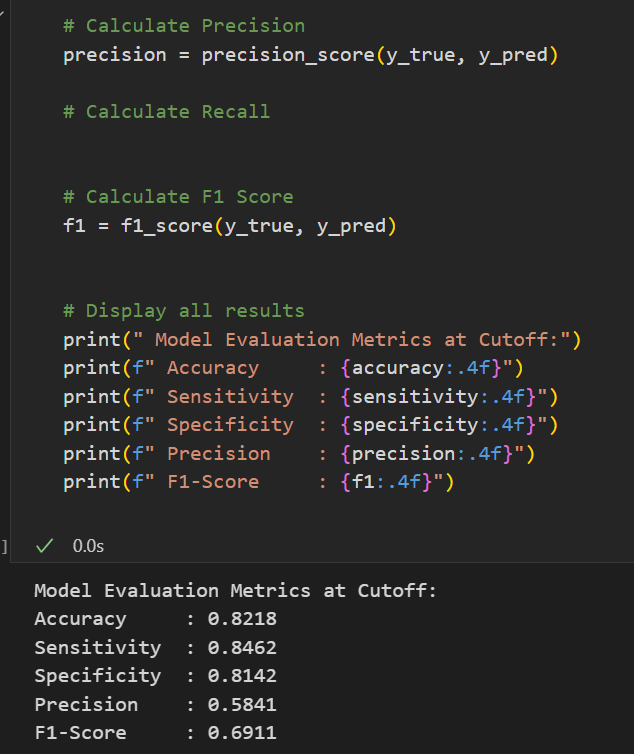
****

****

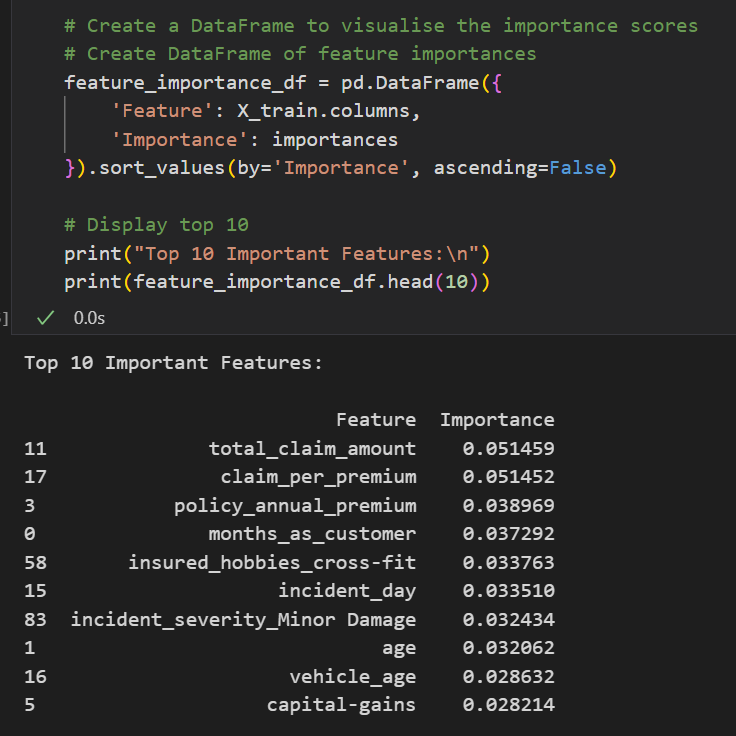
* Intersection of sensitivity and specificity curves = good candidate for threshold
* If sensitivity is too low, you’re missing frauds
* If specificity is too low, you’re flagging too many innocent ones
* Look for a cutoff that balances all three if no one metric is more important.
* If Predicted\_Probability >= 0.4, model predicts fraud (1)
* Else, predicts no fraud (0)
* Now this column is ready for performance evaluation or exporting
  + - **Final model and performance parameters**

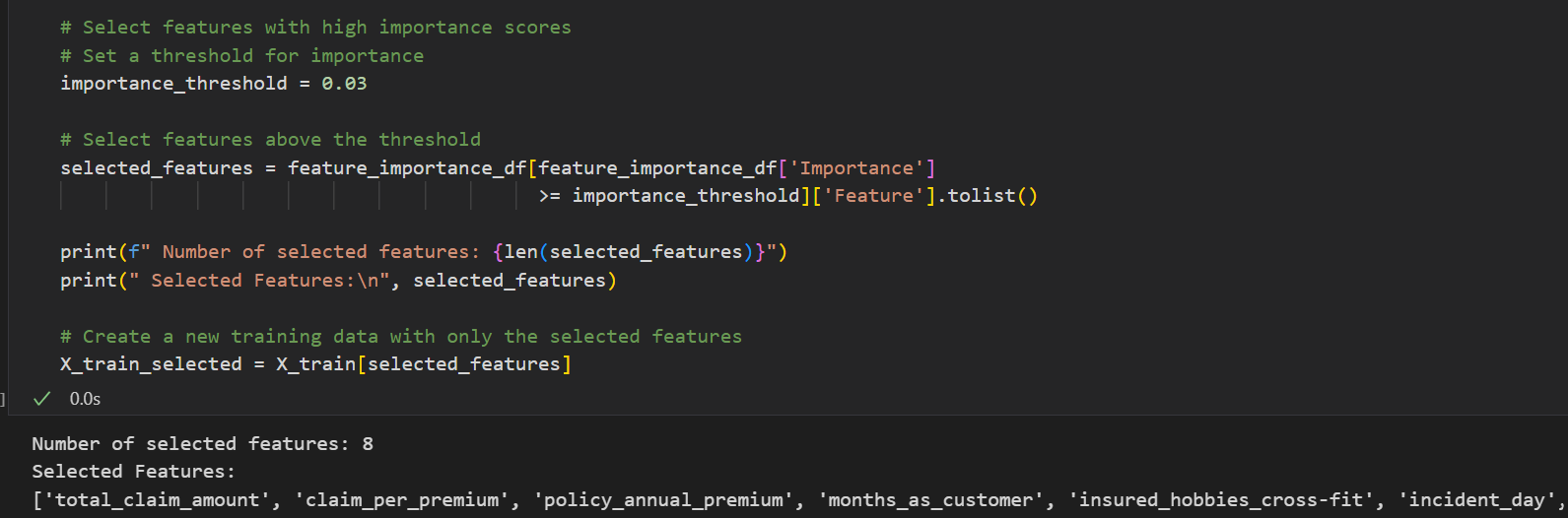
****

****

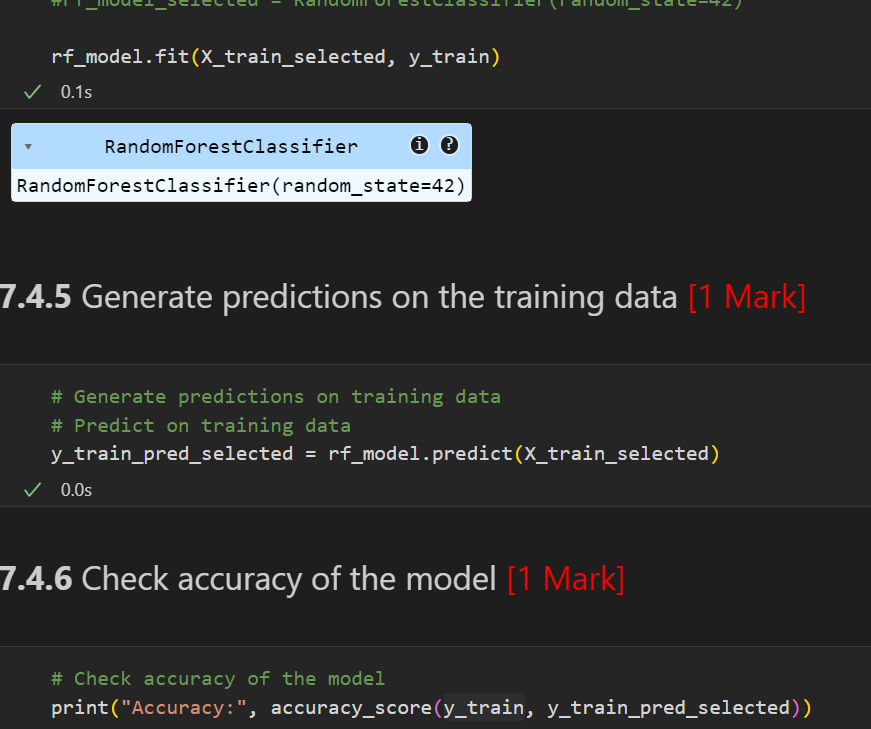
****

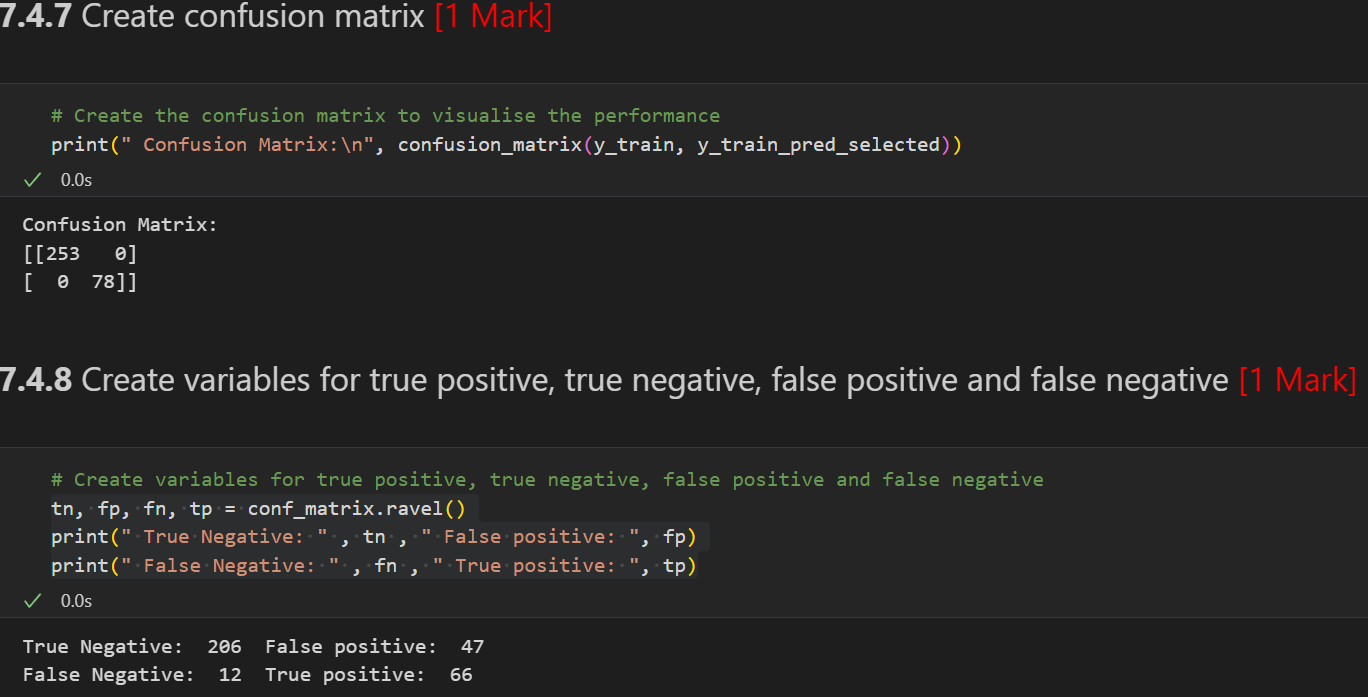
* 1. **Model Random Forest**
     + **Important features selected**

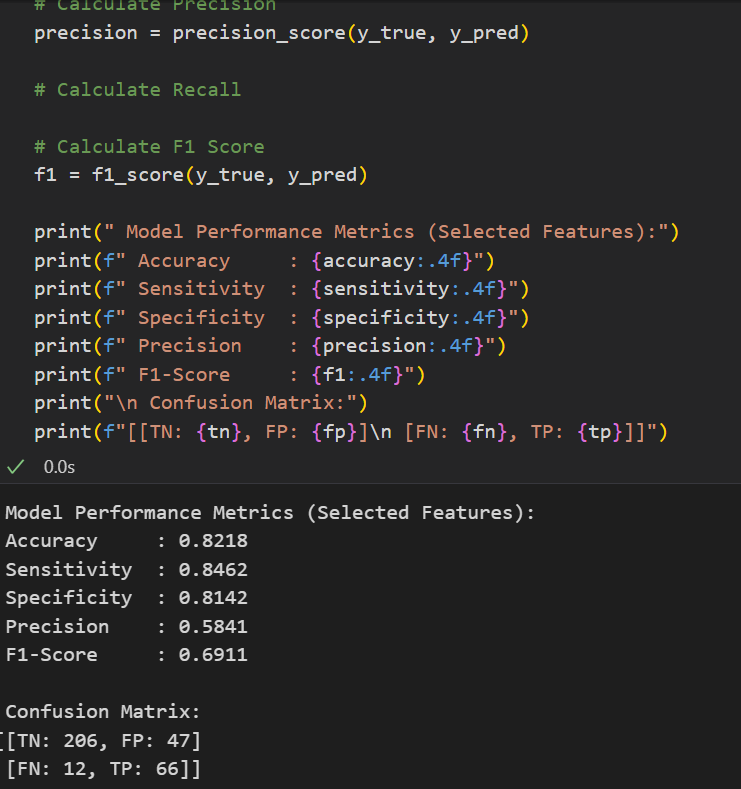
****

****

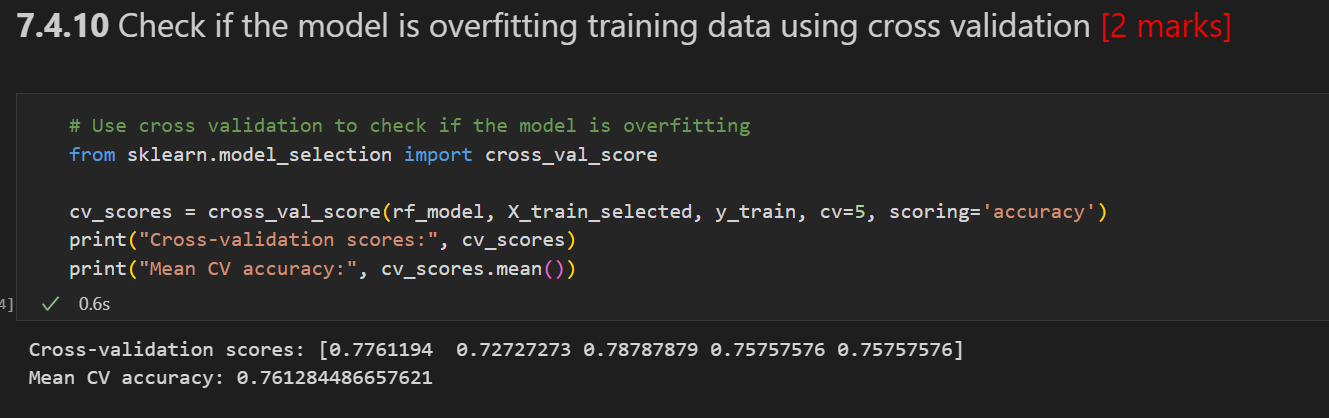
* + - **Model performance parameters**

****

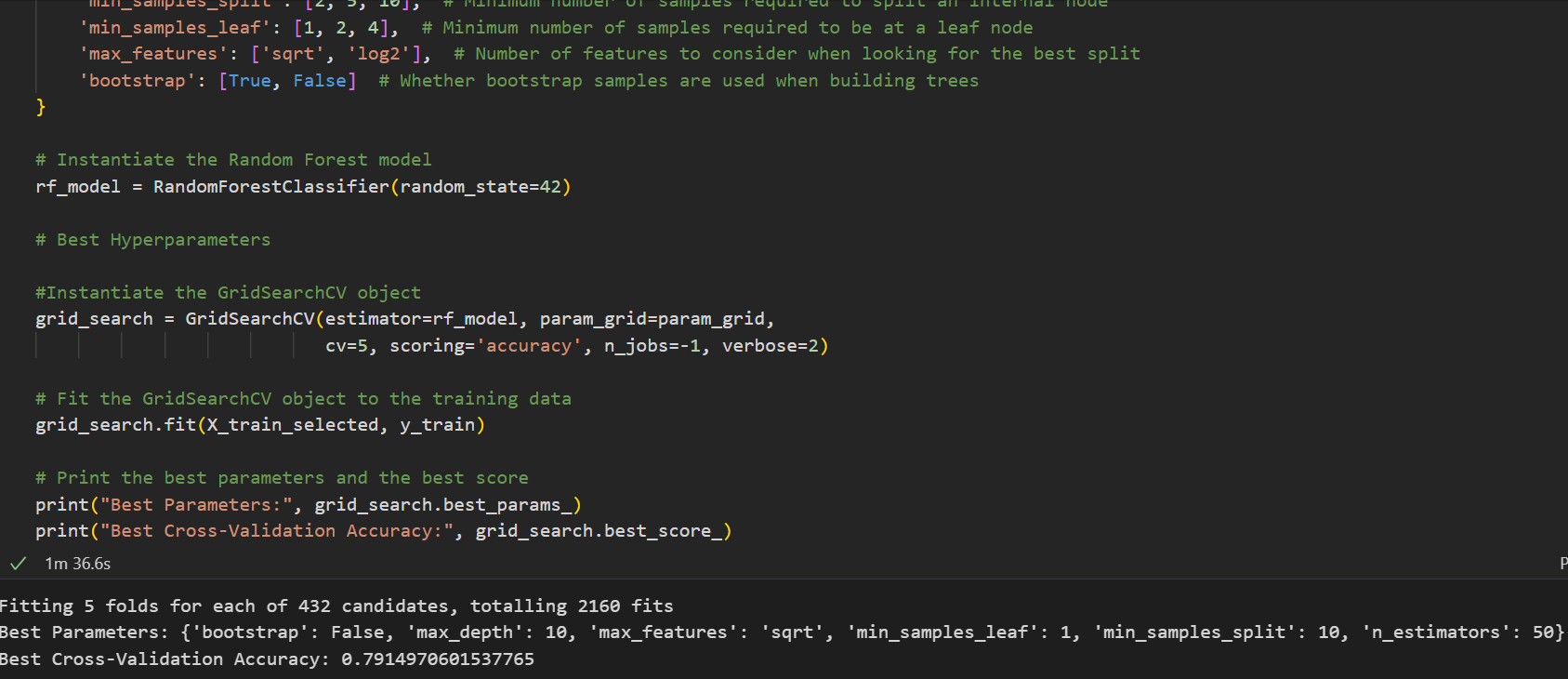
****

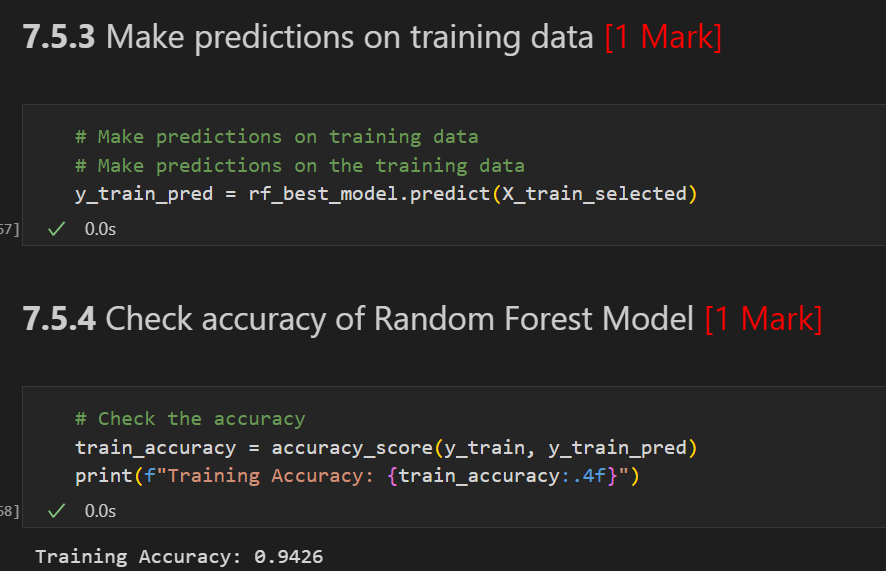
****

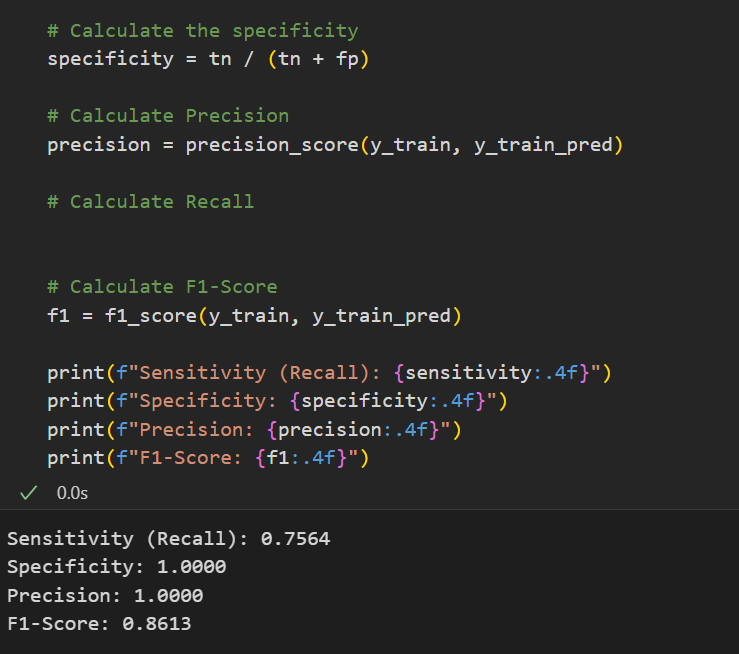
* Sensitivity (Recall) – How well frauds are caught
* Specificity – How well non-frauds are correctly ignored
* Precision – Out of predicted frauds, how many were correct
* F1 Score – Balance between precision & recall
* Accuracy – Overall correctness
  + - **Overfitting**

****

* If training accuracy is much higher than mean cross-validation accuracy, it could indicate overfitting.
* Ideally, both scores should be close — meaning the model generalizes well.
  + - **Hyperparameter**

****

****

****