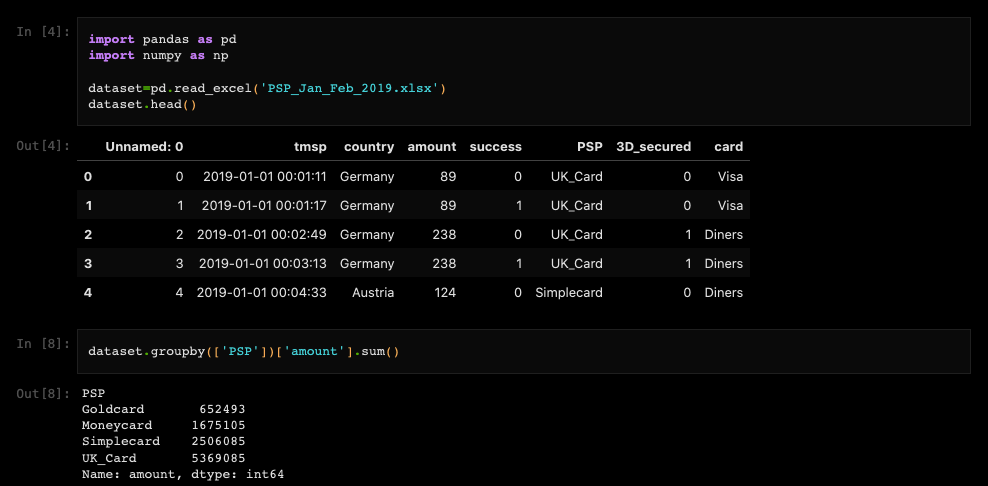
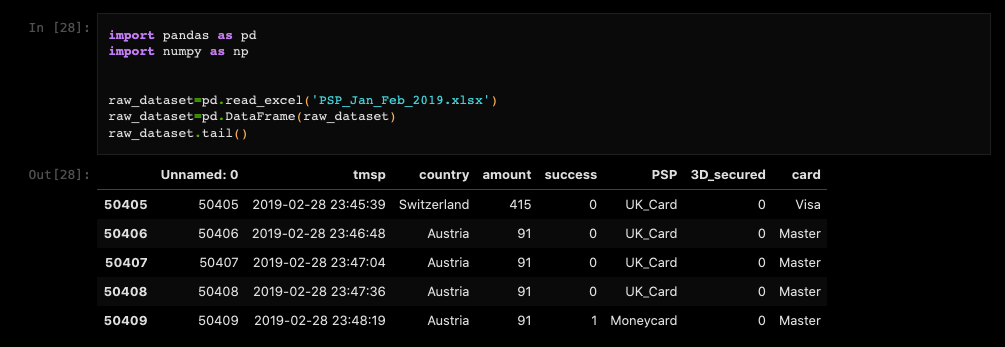
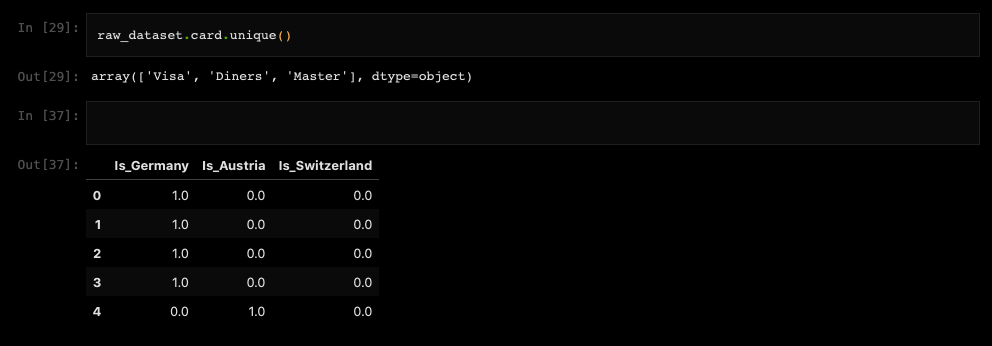
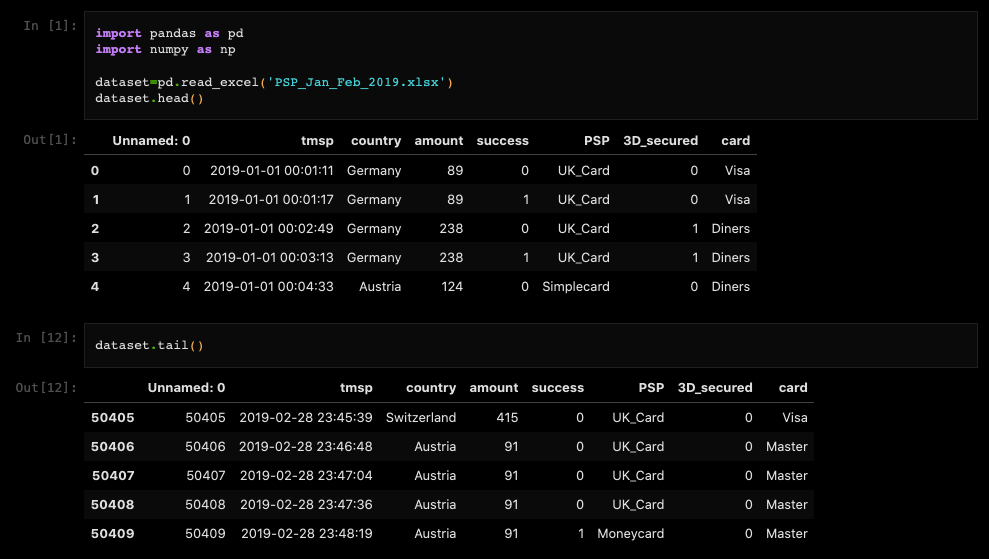
**Explanation of the predictive model with code and the output**

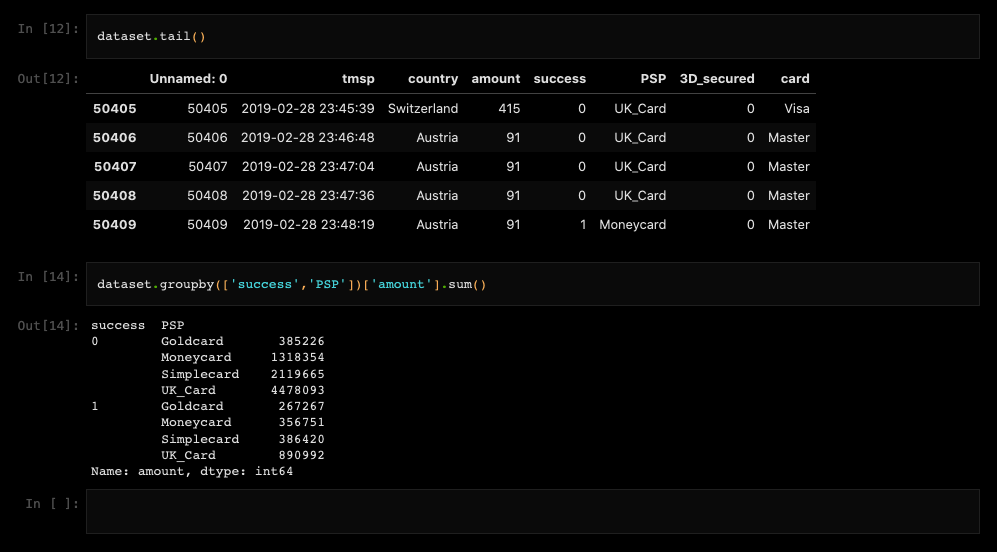
This code uses the pandas library in Python to read an Excel file named 'PSP\_Jan\_Feb\_2019.xlsx' and stores the data in a DataFrame called 'dataset'. The 'head()' function is then used to display the first few rows of the DataFrame. In simple words, the code reads an Excel file, stores the data in a table-like structure called 'dataset', and shows the initial rows of that table.











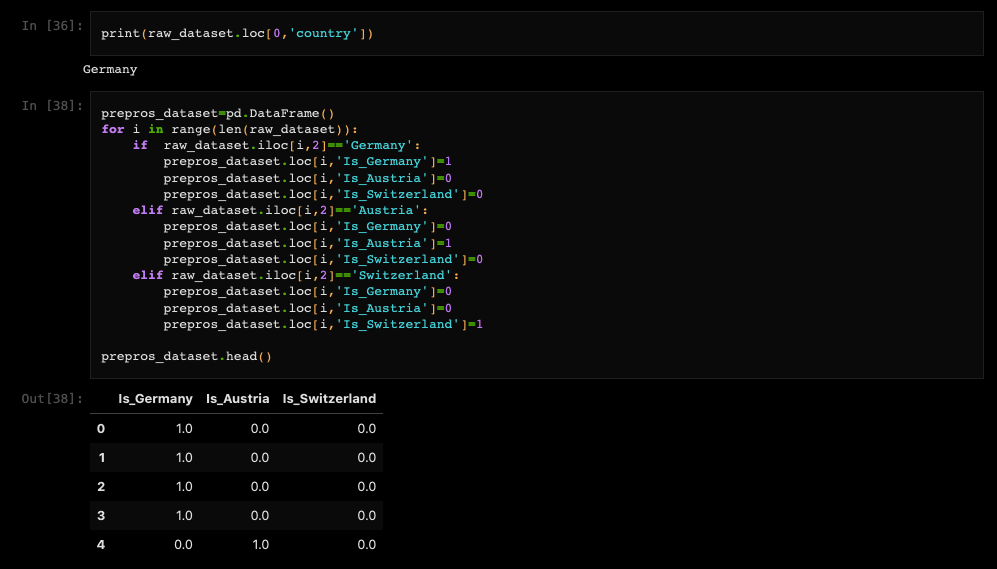
This code creates a new DataFrame called 'prepros\_dataset' and populates it based on the values in the 'raw\_dataset' DataFrame. It checks the value in the third column (index 2) of each row in 'raw\_dataset' and assigns binary values (1 or 0) to corresponding columns in 'prepros\_dataset' based on the country name.

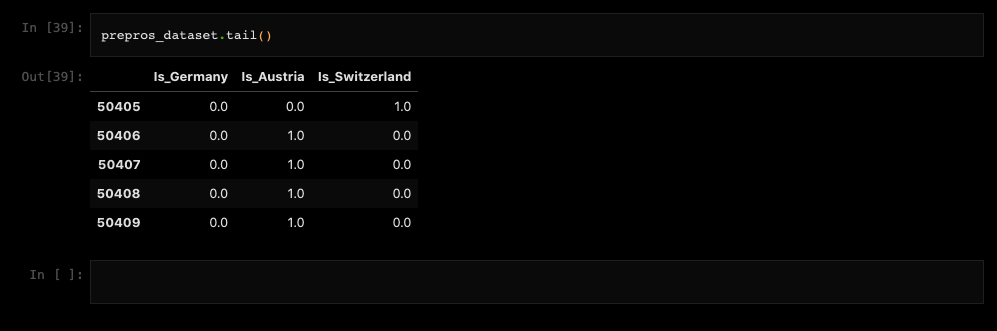
If the country is Germany, it sets the values 'Is\_Germany' to 1 and 'Is\_Austria' and 'Is\_Switzerland' to 0.

If the country is Austria, it sets 'Is\_Austria' to 1 and 'Is\_Germany' and 'Is\_Switzerland' to 0.

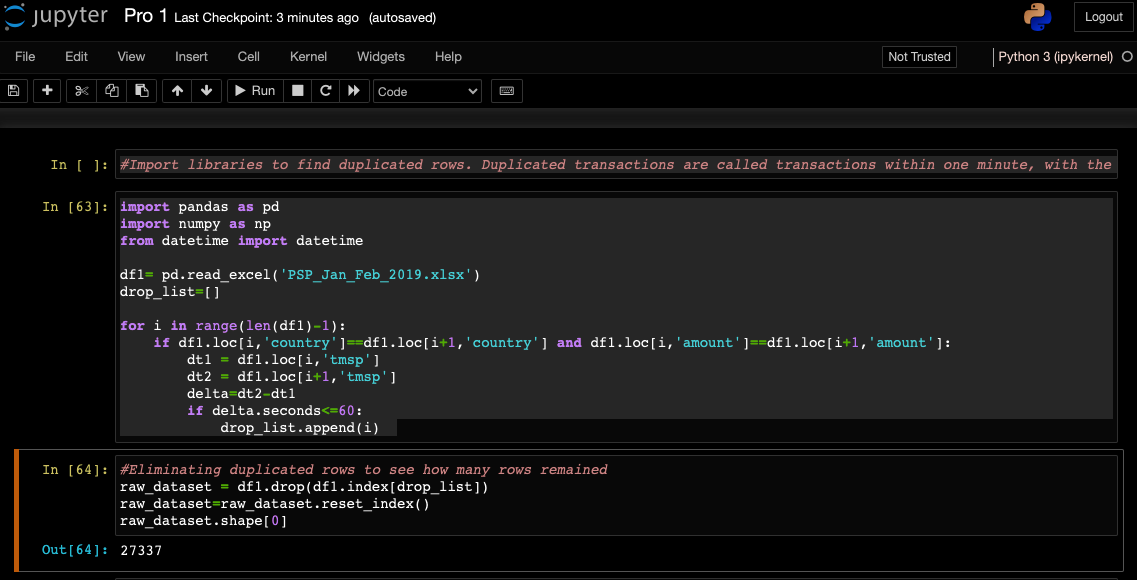
If the country is Switzerland, it sets 'Is\_Switzerland' to 1 and 'Is\_Germany' and 'Is\_Austria' to 0.

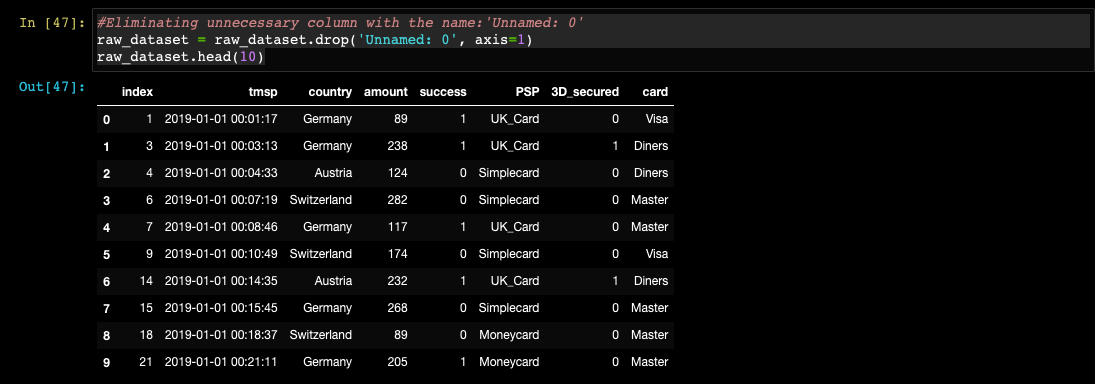
Finally, it displays the first few rows of the 'prepros\_dataset' DataFrame. In simple words, this code creates a new DataFrame where each row represents a country, and the columns indicate whether the country is Germany, Austria, or Switzerland (using binary values). The resulting DataFrame shows the first few rows of this transformed dataset.



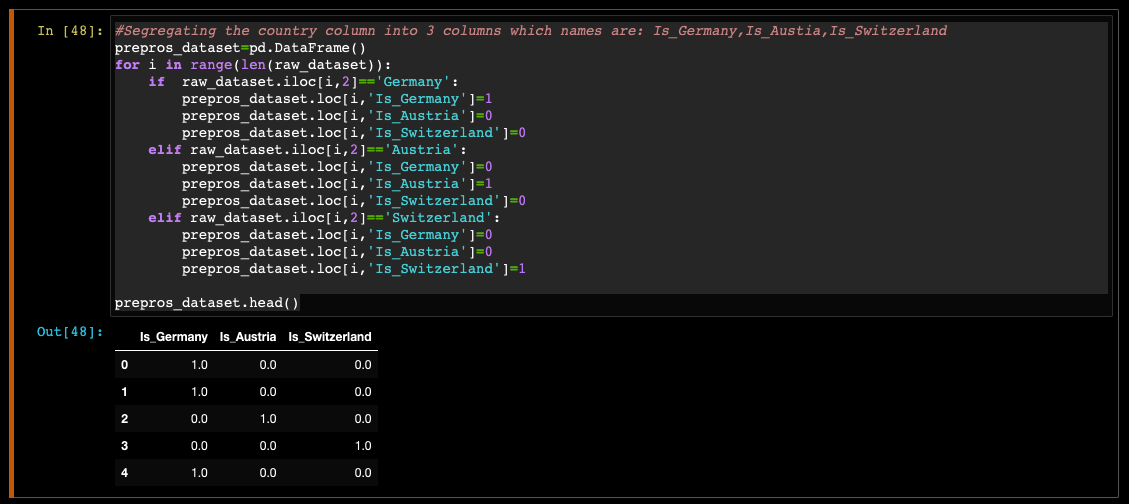


This code reads an Excel file named 'PSP\_Jan\_Feb\_2019.xlsx' using pandas, checks for consecutive rows with the same country and amount values, calculates the time difference between their timestamps, and if the time difference is less than or equal to 60 seconds, it adds the index to the 'drop\_list' to be dropped from the DataFrame 'df1'.

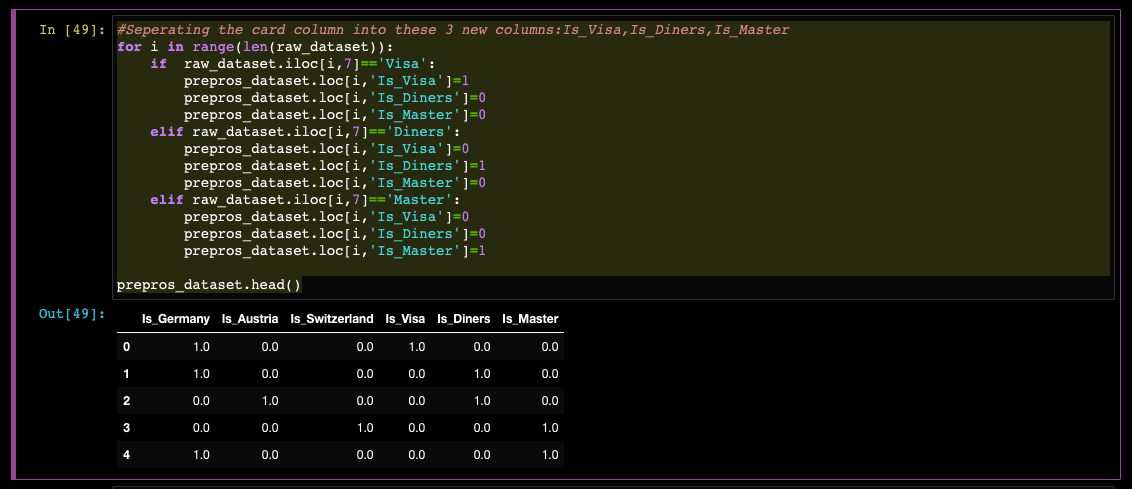




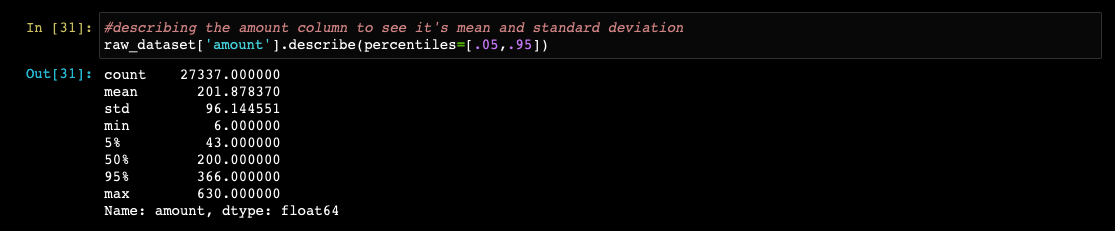
This code creates a new DataFrame called 'prepros\_dataset' and populates it based on the values in the 'raw\_dataset' DataFrame. It segregates the 'country' column into three separate columns ('Is\_Germany', 'Is\_Austria', 'Is\_Switzerland') by assigning binary values (1 or 0) to indicate the respective country for each row in the dataset. The resulting DataFrame shows the first few rows of this transformed dataset.



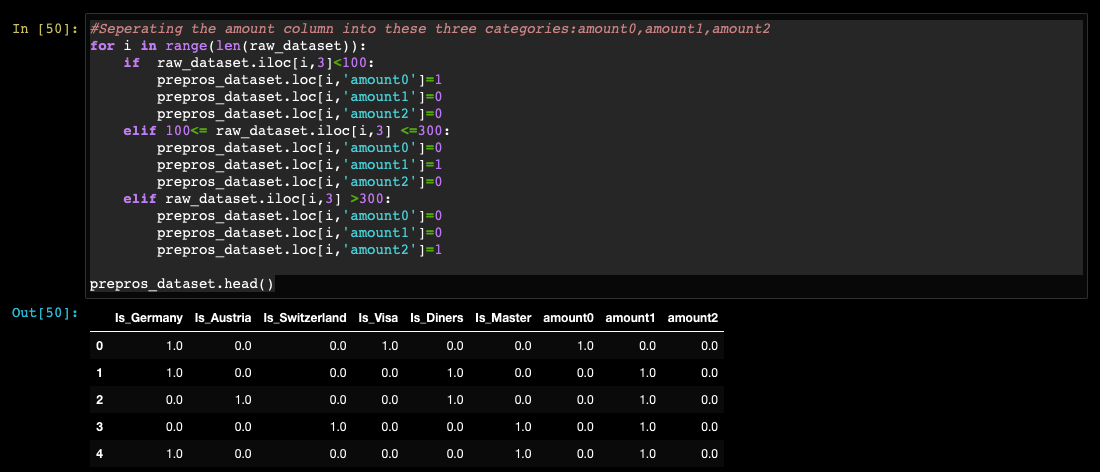
This code creates new columns in the 'prepros\_dataset' DataFrame named 'Is\_Visa', 'Is\_Diners', and 'Is\_Master'. It assigns binary values (1 or 0) to indicate the respective card type (Visa, Diners, or Master) based on the values in the 'card' column of the 'raw\_dataset'. The resulting DataFrame shows the first few rows of this transformed dataset.



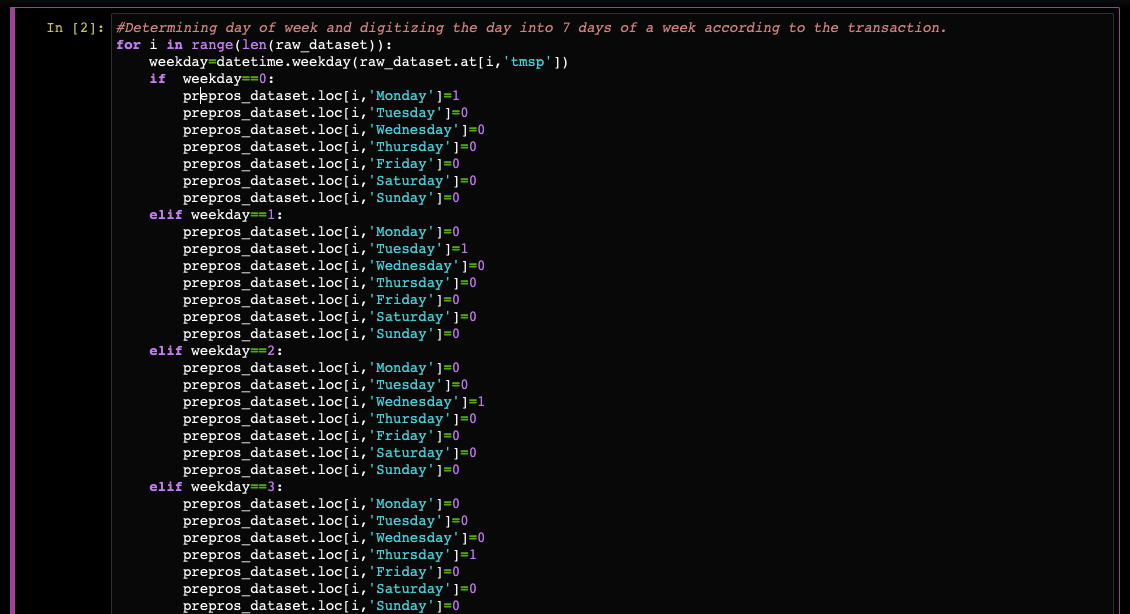
This code calculates descriptive statistics for the 'amount' column in the 'raw\_dataset', including the mean and standard deviation. Additionally, it provides percentiles (specifically, the 5th and 95th percentiles) to analyse the distribution of the data.

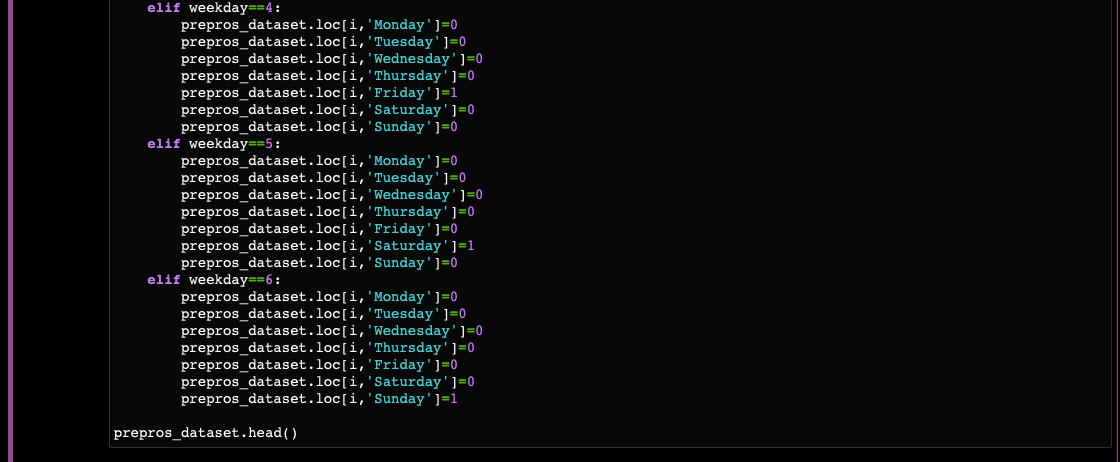


This code creates new columns in the 'prepros\_dataset' DataFrame named 'amount0', 'amount1', and 'amount2'. It assigns binary values (1 or 0) to indicate the respective category for each row in the 'raw\_dataset' based on the value in the 'amount' column. The categories are determined by different ranges: less than 100, between 100 and 300 (inclusive), and greater than 300. The resulting DataFrame shows the first few rows of this transformed dataset.

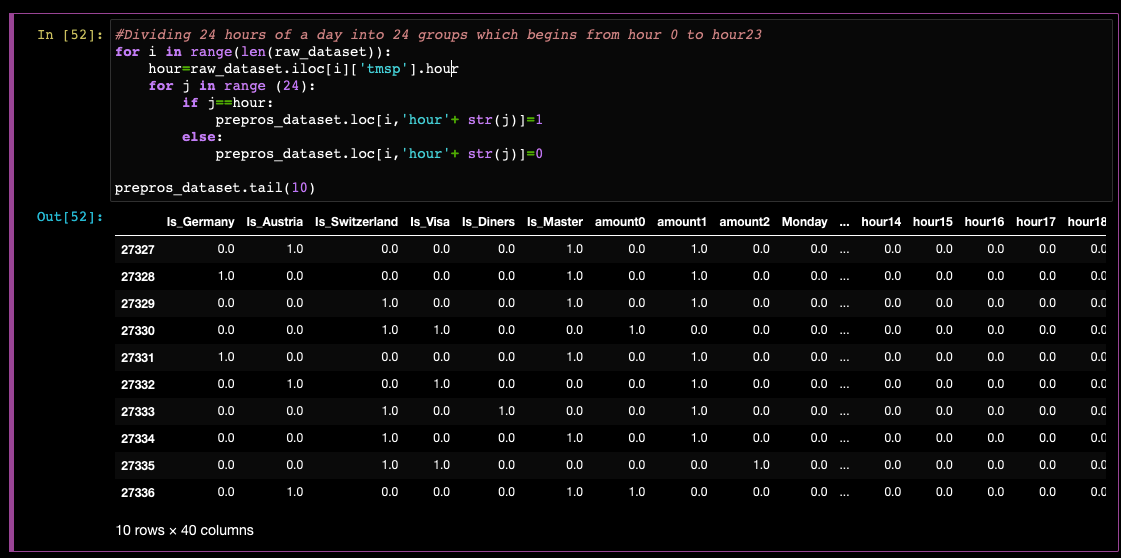


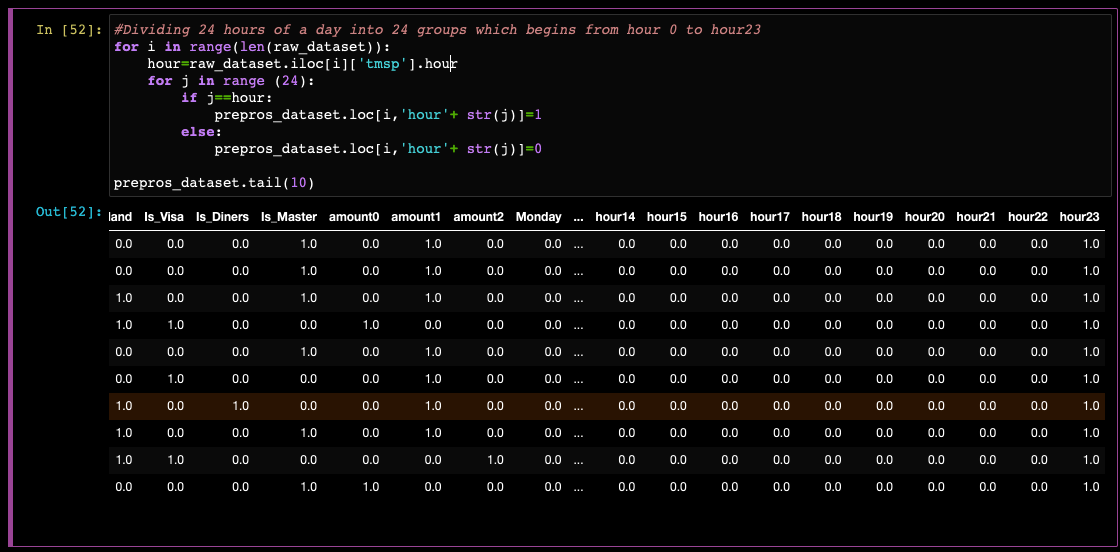
This code creates new columns in the 'prepros\_dataset' DataFrame representing the days of the week ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'). It determines the day of the week for each transaction's timestamp in the 'raw\_dataset' and assigns a binary value (1 or 0) to the corresponding day column to indicate which day of the week the transaction occurred. The resulting DataFrame shows the first few rows of this transformed dataset.

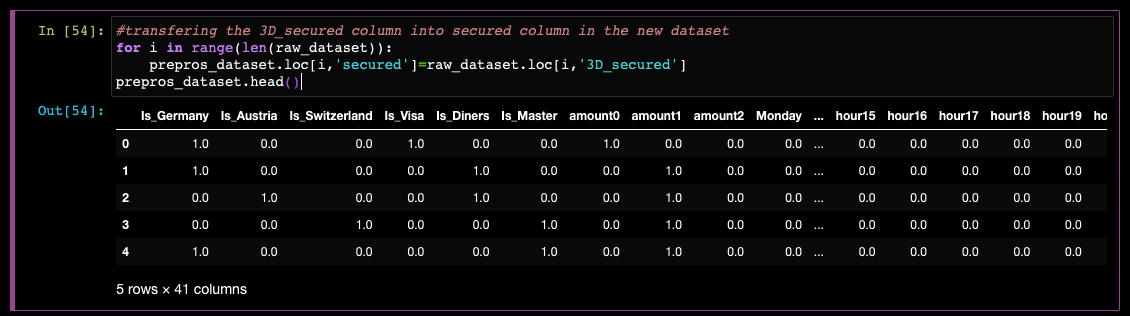




This code creates 24 new columns in the 'prepros\_dataset' DataFrame, representing each hour of the day from 0 to 23. It determines the hour of the transaction's timestamp in the 'raw\_dataset' and assigns a binary value (1 or 0) to the corresponding hour column to indicate the hour in which the transaction occurred. The resulting DataFrame shows the last 10 rows of this transformed dataset.



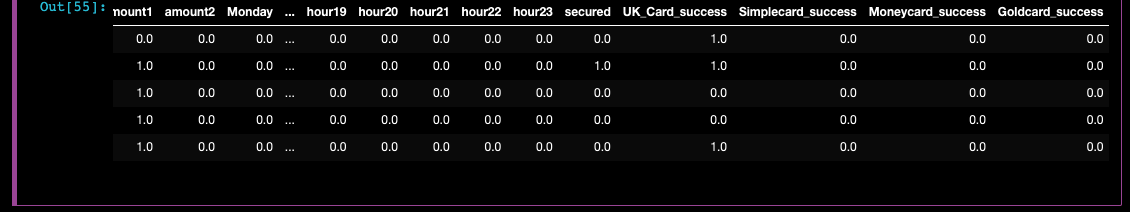




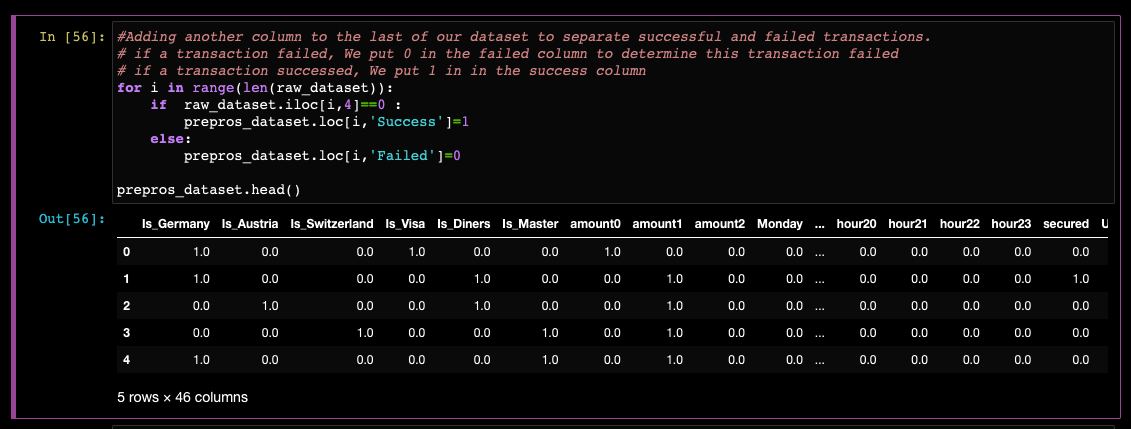
This code defines four columns in the 'prepros\_dataset' DataFrame as model targets. Each column corresponds to a specific Payment Service Provider (PSP). If a transaction is successful for a particular PSP, the corresponding column is set to 1, and the other PSP columns are set to 0. If a transaction fails, all four columns are set to 0. The resulting DataFrame shows the first few rows of this transformed dataset.

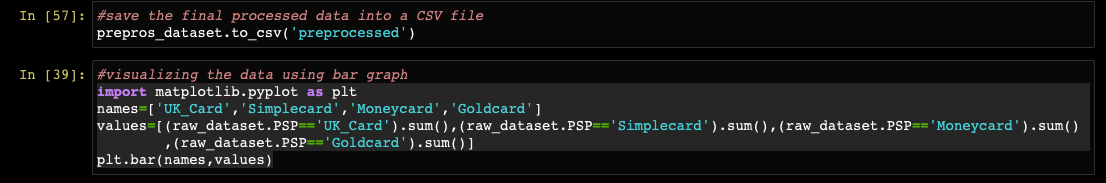


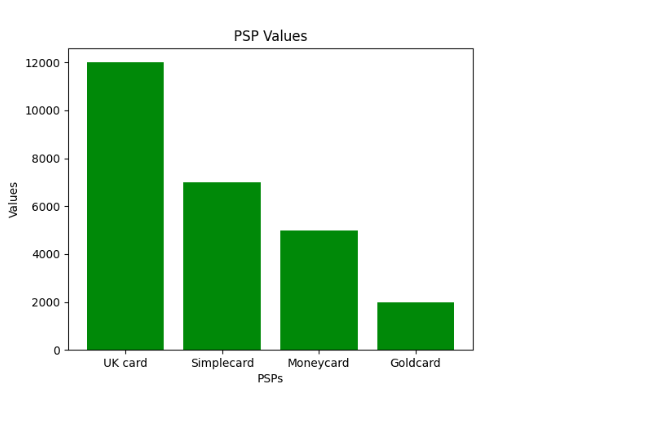




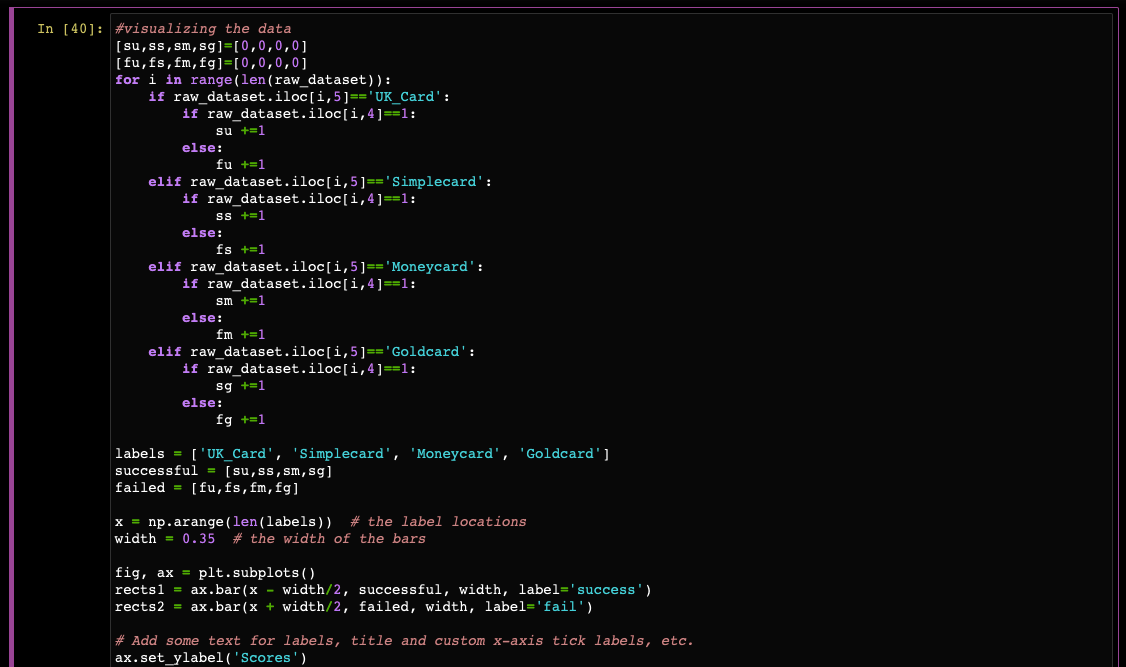
This code adds two additional columns to the 'prepros\_dataset' DataFrame. The 'Success' column is set to 1 for successful transactions, while the 'Failed' column is set to 0 to indicate failed transactions. The resulting DataFrame shows the first few rows of this updated dataset, indicating the success or failure of each transaction.

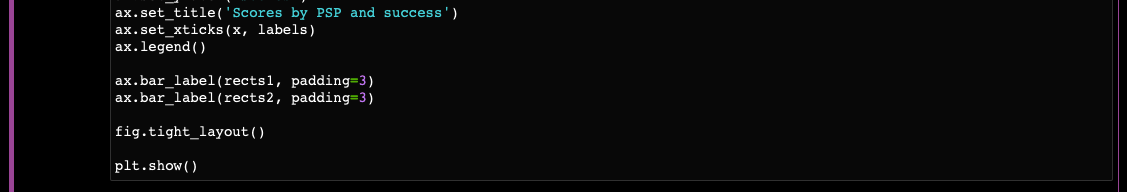


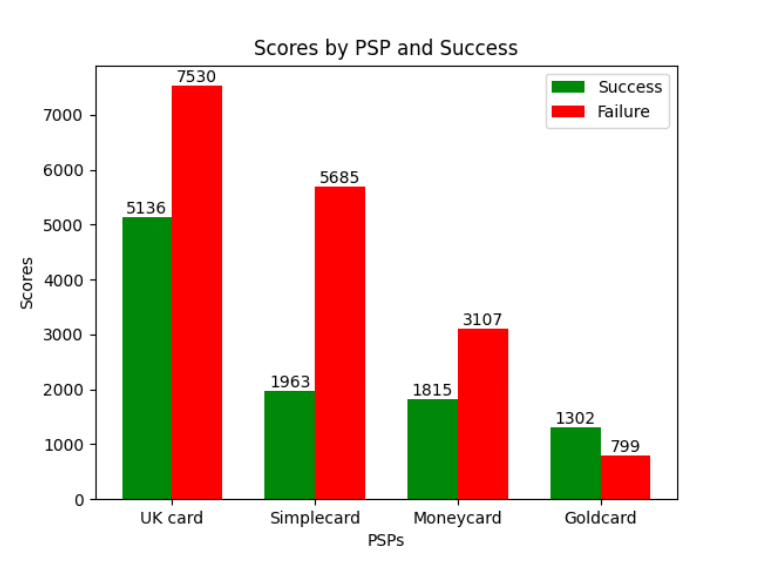


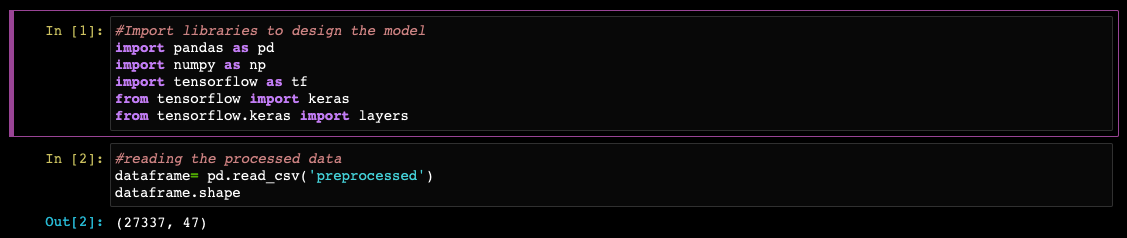


This code visualizes the data by creating a bar chart using matplotlib. It calculates the counts of successful and failed transactions for each Payment Service Provider (PSP) and stores them in separate variables. The code then plots the counts using labeled bars for each PSP, distinguishing between successful and failed transactions. The resulting bar chart provides a visual representation of the success and failure rates across different PSPs.

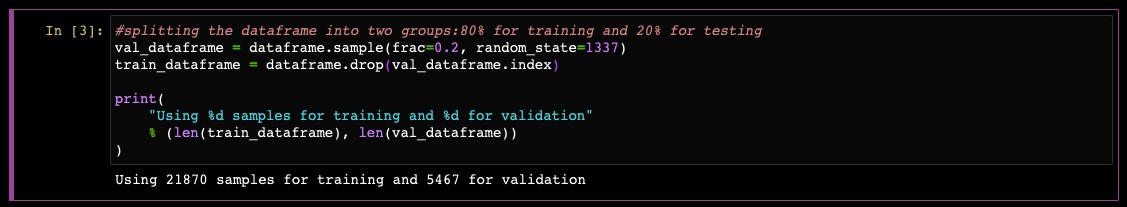




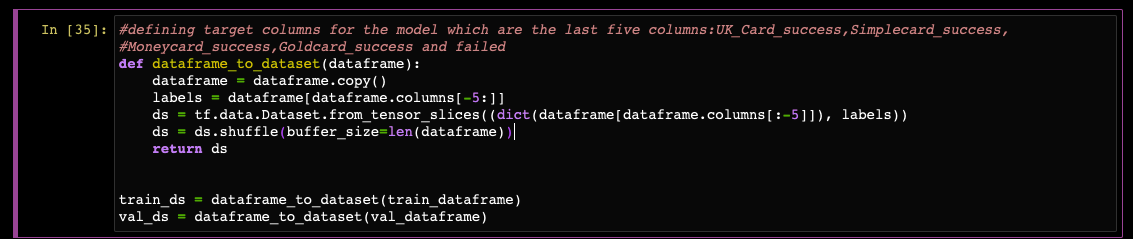


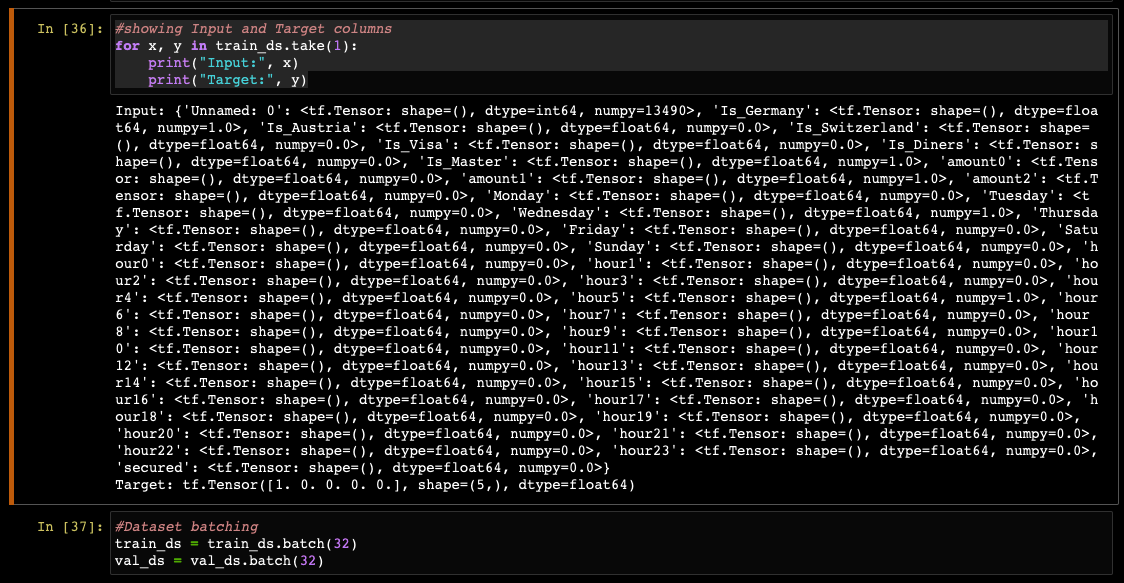


This code splits the original DataFrame into two groups: a training set and a validation set. It uses a random sampling technique to select 20% of the data for the validation set, while the remaining 80% is assigned to the training set. The random\_state parameter ensures reproducibility of the split. The code then prints the number of samples allocated for training and validation, respectively.

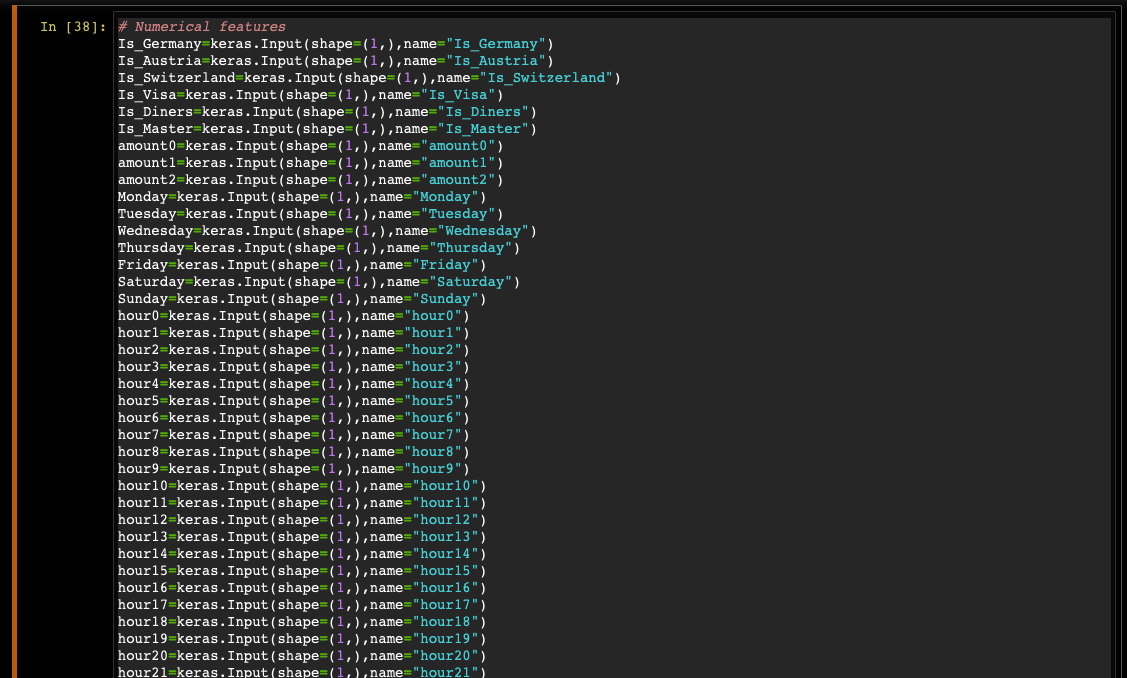


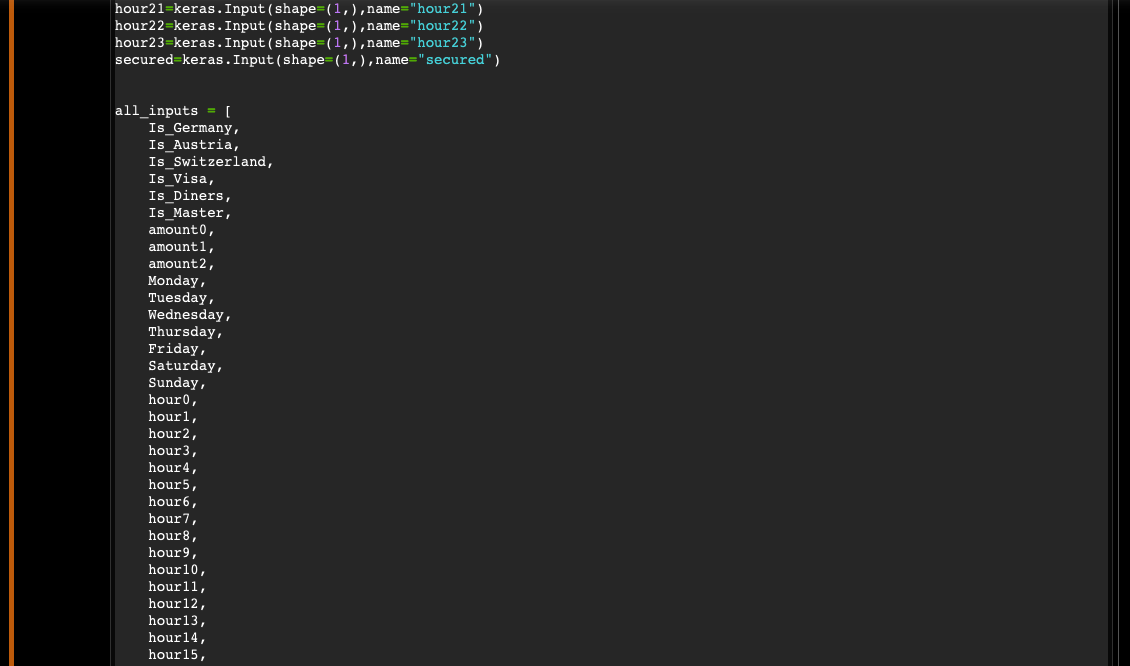
This code defines a function called 'dataframe\_to\_dataset' that converts a DataFrame into a TensorFlow Dataset. It copies the DataFrame, separates the last five columns (target columns) as labels, creates a TensorFlow Dataset from the input features and labels, and shuffles the dataset. The function is then used to create training and validation datasets from the corresponding DataFrames. Finally, it displays a sample input and target columns from the training dataset.

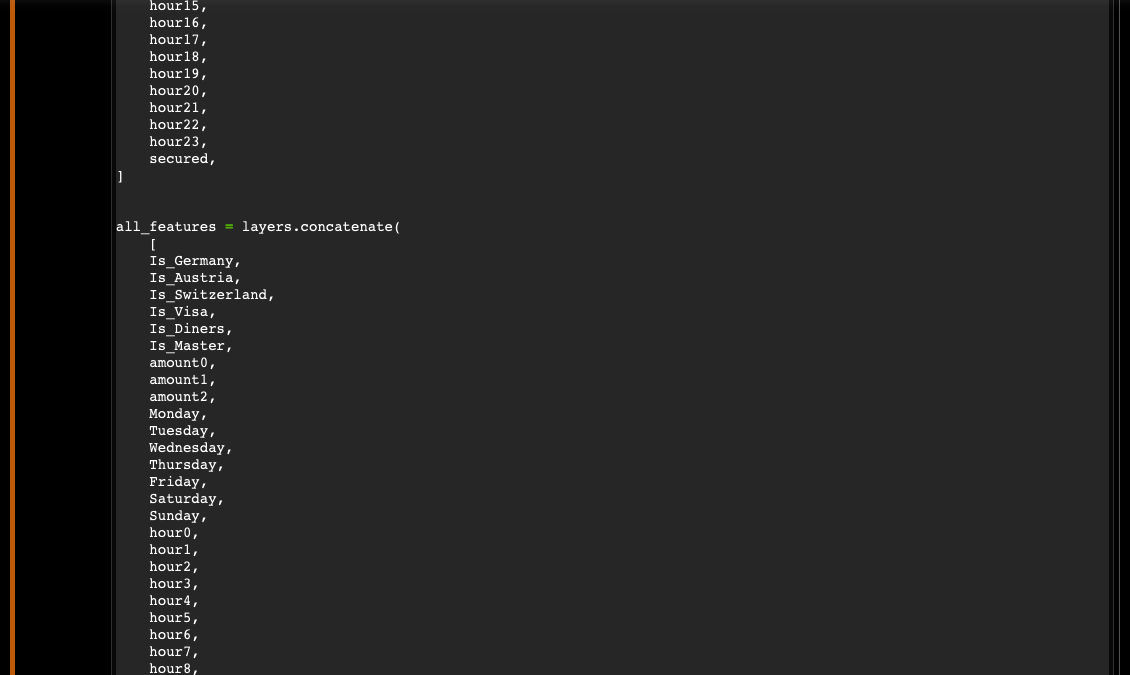


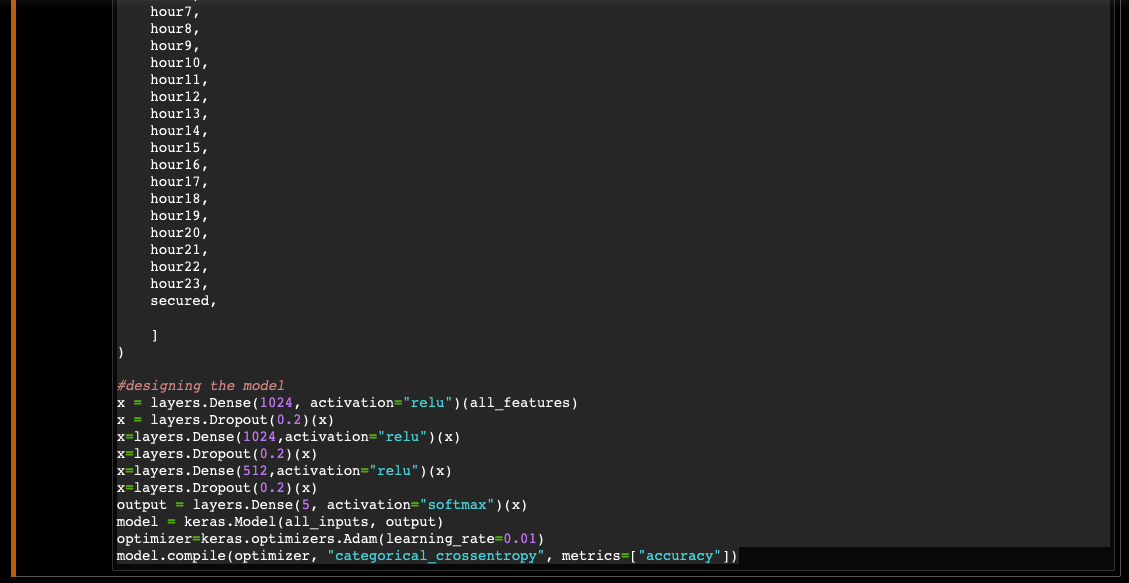


This code defines the inputs and features for a deep learning model using the Keras library. It creates input layers for various numerical features such as country, card type, amount, day of the week, hour of the day, and security information. It then concatenates all the input layers into a single feature layer. The model architecture consists of multiple dense layers with dropout regularization. The output layer has 5 nodes representing different target categories. The model is compiled with an optimizer, loss function, and evaluation metrics.

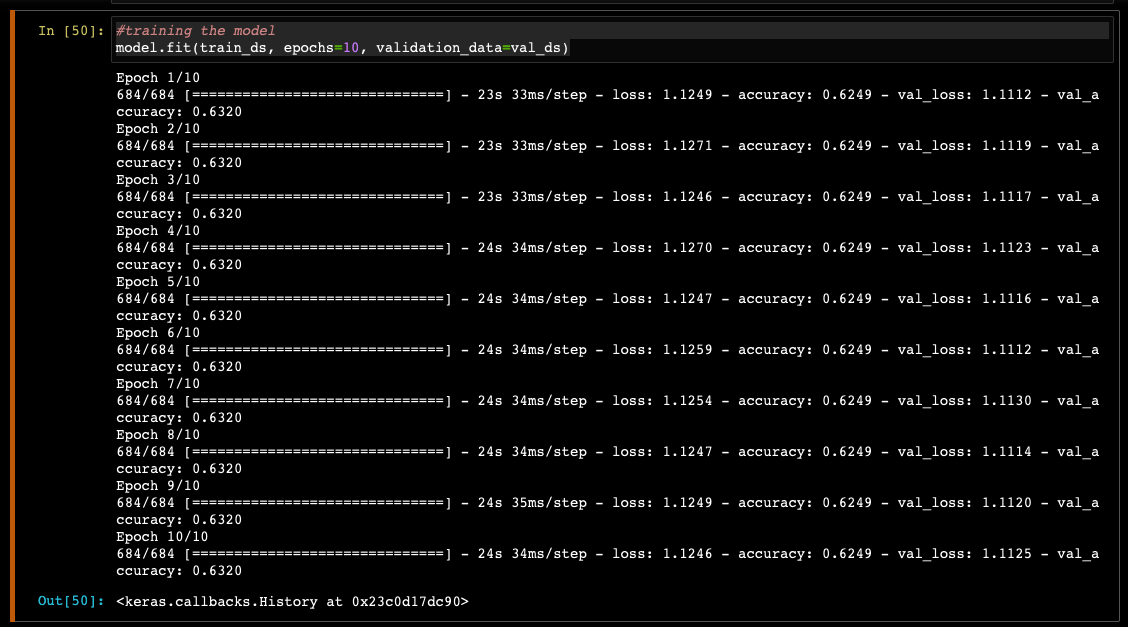




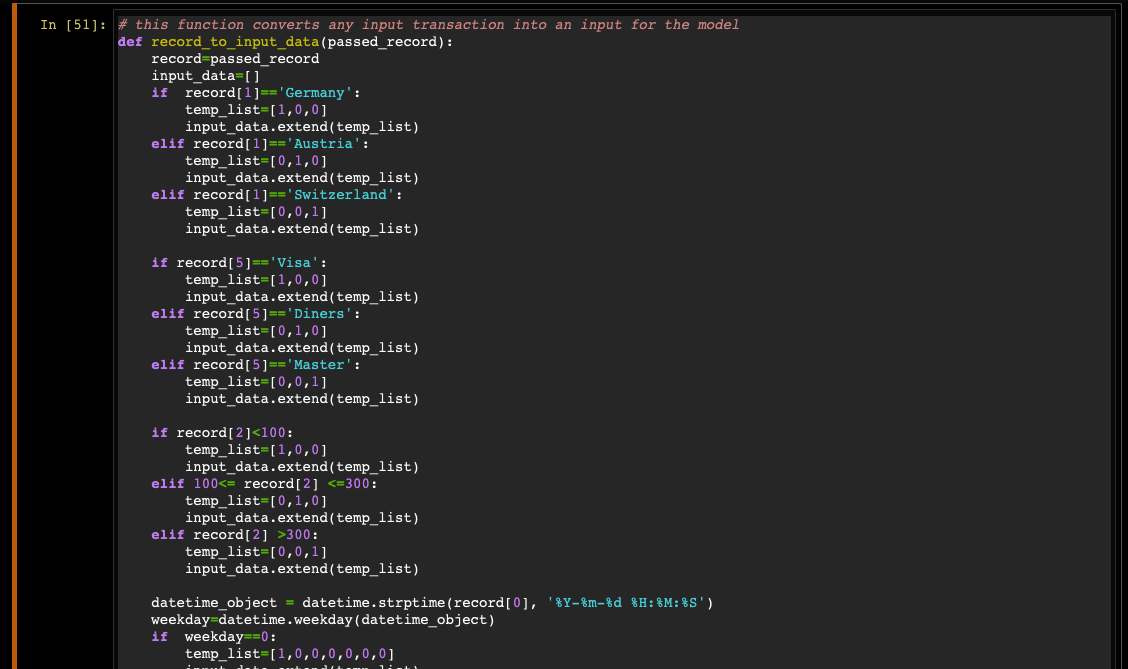


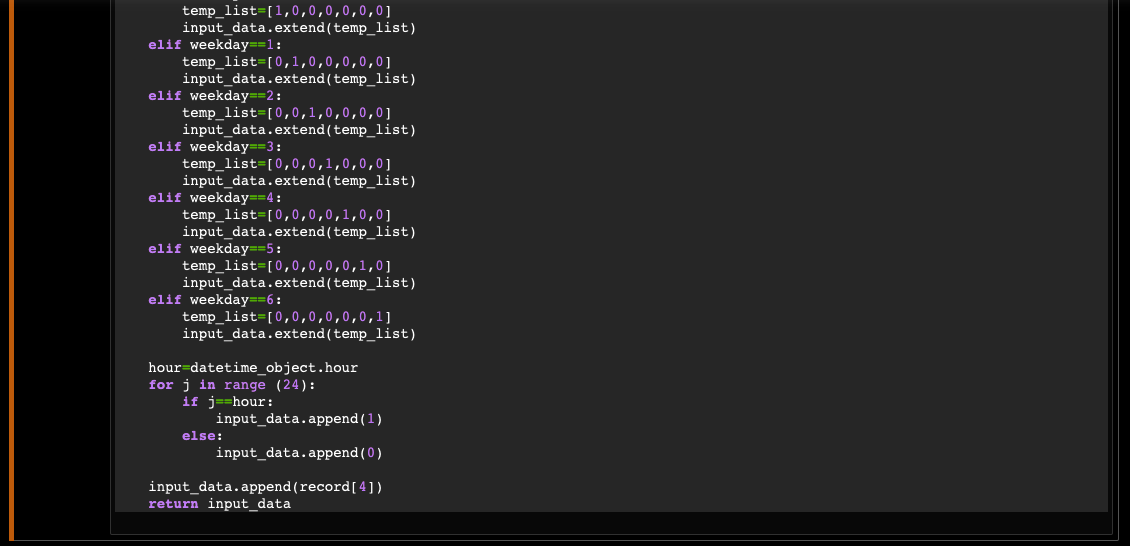


This code trains the defined model using the training dataset (train\_ds) for 10 epochs. It validates the model's performance on the validation dataset (val\_ds) during training.

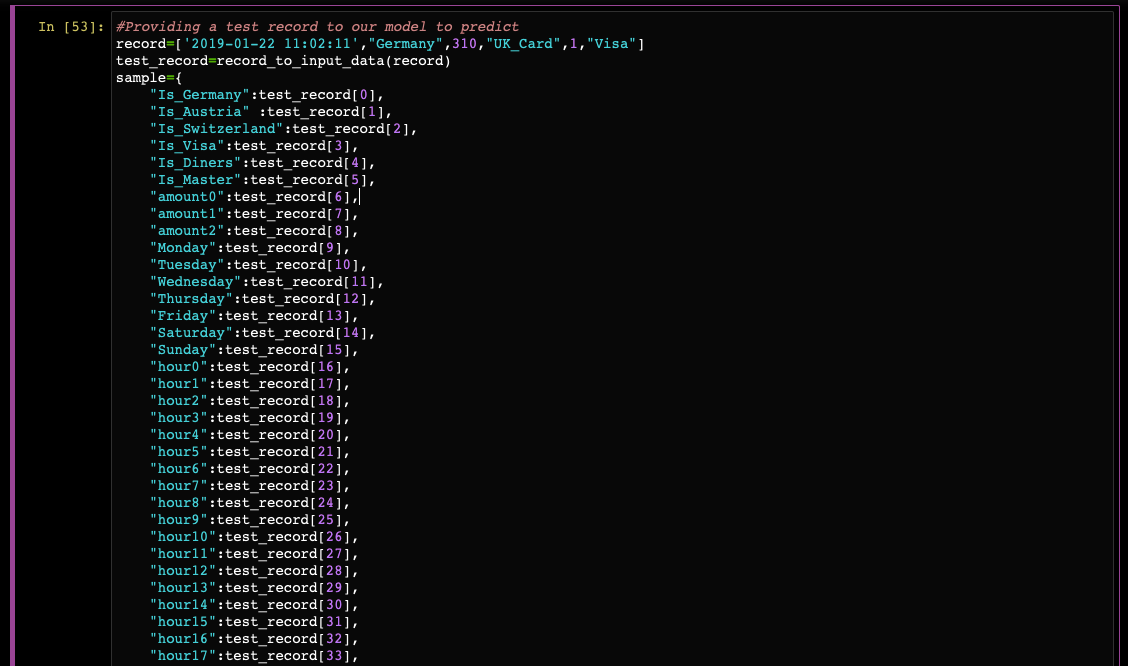


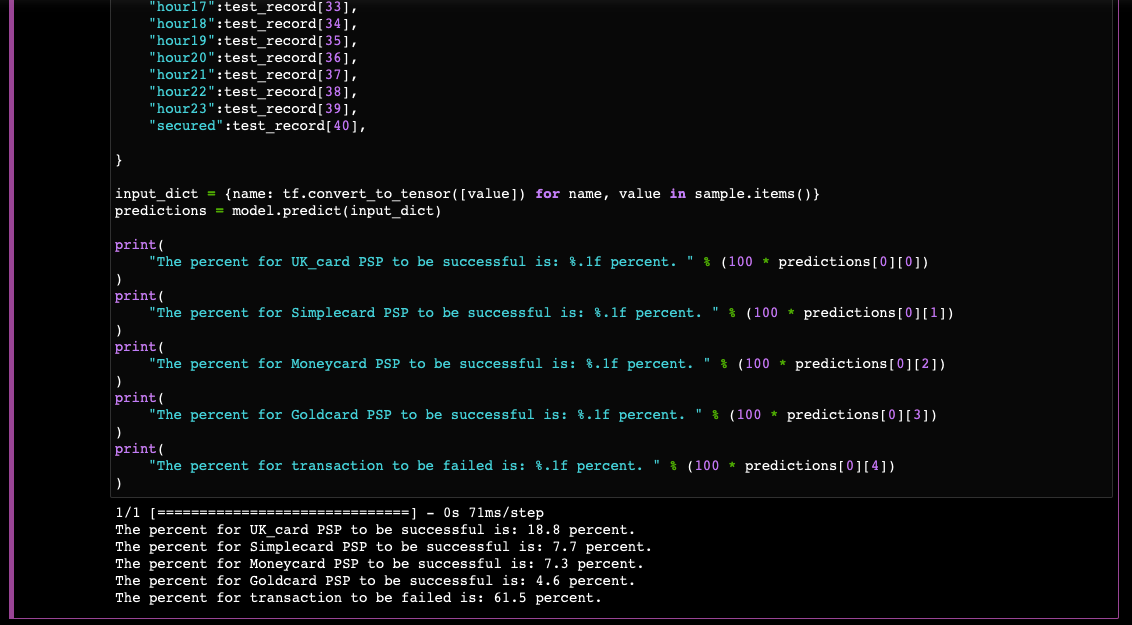
This code defines a function called record\_to\_input\_data that takes a transaction record as input and converts it into an input data format suitable for the model. The function extracts various attributes from the record, such as country, card type, amount, date, and hour, and maps them to specific numerical representations. It constructs an input data list based on these mappings and returns it.





This code provides a test record to the model to predict the success probabilities for different Payment Service Providers (PSPs) and the failure probability of the transaction. It calls the record\_to\_input\_data function to convert the test record into the required input format for the model. The input data is then constructed as a dictionary with the corresponding feature names and values. The model predicts the probabilities using the predict method and prints the success and failure probabilities for each PSP and the overall transaction.





Based on the predicted probabilities for the test transaction, the code selects the PSP with the highest probability of success. It multiplies the probabilities by 100 to convert them into percentages and rounds them to two decimal places. It then finds the index of the maximum probability and determines the corresponding PSP. The code prints the recommendation for the PSP selection based on the maximum probability.

