Explore and get insights of the given dataset for the Client requirement.

As the given dataset are defining it is a large multinational corporation, the financial team of Atlantis validates and checks thousands of bills and classifies them to the different bill categories according to the below information of the bill:

1. Company Name

2. Financial Department

3. Financial Account Group

4. Vendor Name

5. The Bill amount for the past 4 months.

The Bills data with the required attributes is provided and the Categories are provided for some bills in the last column, for some bills, Category is blank. This will form the required data set for this use case. I have to split the dataset into two where it contains value at Category columns and another set has Category empty.

**1.Train\_Data.xlsx**

**2.Test\_Data. Xlsx**

**Work flow for the given dataset:**

* Importing libraries
* Fetching the dataset
* Creating the dependent variable class
* Extracting features and do Feature Engineering
* Train-Test dataset splitting
* Feature scaling
* Training the model
* Calculating the model score using the metric deemed fit based on the problem
* Saving the model for future use

**Importing libraries:**

**Pandas**: One of the most popular libraries for data manipulation and storage. This is used to read/write the dataset and store it in a dataframe object.

**Numpy**: The library used for scientific computing. Here we are using the function vectorise for reversing the factorization of our classes to text.

**Sklearn**: The library is used for a wide variety of tasks, i.e. dataset splitting into test and train, training the random forest, KNN, Decision tree classifier and creating the confusion matrix.

**Creating the independent and dependent variable class**

**Dependent Class:**

We are basically converting Category column values in the form of Dictionary like {0: 'X0', 1: 'X1', 2: 'X2'…}

This is an essential step as the scikit-learn's algorithms can't predict text it can only predict numbers. Like

**Independent Class:**

Also, we need to store the factor conversions to remember what number is substituting the text. For the independent features like Vendor name, Financial Department, Company Name etc.

The code below will perform the following:

* Use pandas factorize function to factorize the species column in the dataset. This will create both factors and the definitions for the factors.
* Store the factorized column as category.
* Store the definitions for the factors.
* Show the first five rows for the species column and the definitions array.

**Extracting features and do Feature Engineering and Exploratory data analysis:**

* If the data is missing means, using mean or median to fill the dataset.
* Analysis each every feature with the dependent feature.
* It a way of visualizing, summarize and interpreting the information and find the insights are drawn for bivariate, univariate and multivariate analysis.

**Train-Test Data Splitting**

* We need to split the training data into 80% and 20%. Here the model learns efficiently and learning rate also good.
* Also, the reason for such high number of test case percentages is due to fewer numbers of rows for the model.
* The below code uses the prebuilt function 'train\_test\_split' in a sklearn library for creating the train and test arrays for both independent and dependent variable.
* We are not going to create cross validation datasets, as they are used when hyper parameter training is involved.

**Feature Scaling**

* It helps the algorithm quickly learn a better solution to the problem.
* We will use a standard scaler provided in the sklearn library.
* It subtracts the mean value of the observation and then divides it by the unit variance of the observation.

***We will perform the following steps:***

* Transform train feature dataset (X\_train) and fit the scaler on train feature dataset.
* Use the scaler to transform test feature dataset (X\_test).

**Feature Importance:**

* Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable.

**Problem Statement:**

* Here it is classification problem with structured data so it deals with following any one algorithms,
* Many algorithms used for binary classification can be used for multi-class classification.

Popular algorithms that can be used for multi-class classification include:

* K-Nearest Neighbors.
* Decision Trees.
* Naive Bayes.
* Random Forest.

This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

**One-vs-Rest**: Fit one binary classification model for each class vs. all other classes.

**One-vs-One:** Fit one binary classification model for each pair of classes.

* The model which performed well was the Random Forest algorithm and it is essentially trains multiple decision-trees and averages their collective decisions together.
* It gives accuracy nearly **97%** Compares with others.

**Hyper parameter Tuning:**

* Hyper parameter tuning is a process of tuning some of the parameter which are really used to get more weight for the accuracy
* It is the internal coefficients or weights for a model found by the learning algorithm.
* The more hyper parameters of an algorithm that you need to tune, the slower the tuning process. Therefore, it is desirable to select a minimum subset of model hyper parameters to search or tune.

***n\_estimators=[int(x) for x in np.linspace(100,1200,12)]***

***## Good values might be a log scale from 10 to 1,000.***

***max\_features= ['auto','sqrt']***

***max\_depth=[int(x) for x in np.linspace(5,30,6)]***

***min\_samples\_split= [2,4,6,8,10,15]***

***min\_samples\_leaf=[1,2,5,10]***

**Model Save:**

* Model is saved by using pickle and it is used for future Category predictions.