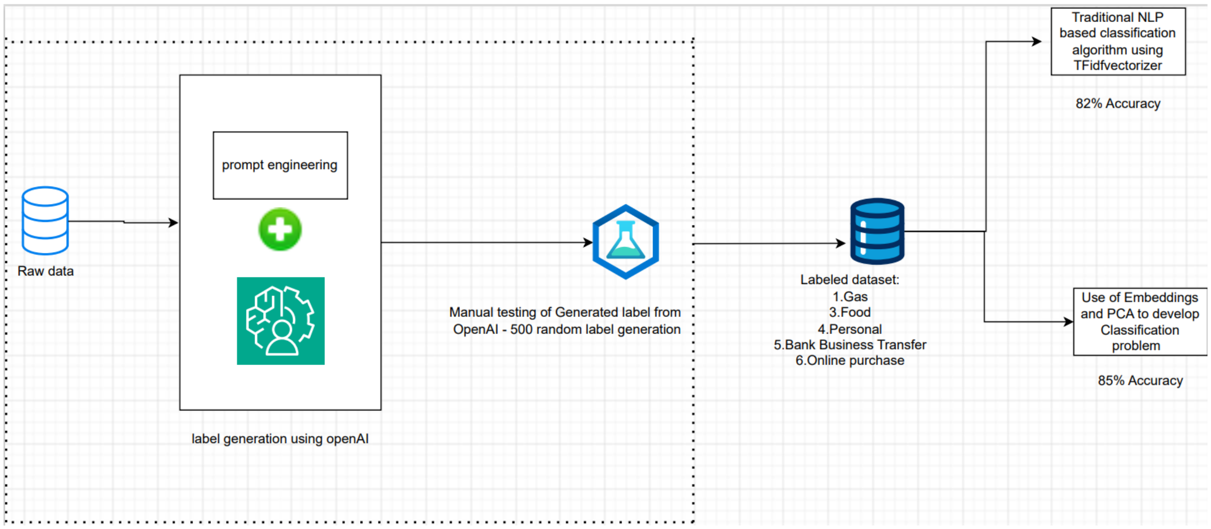
This Document presents the steps for generating and utilising the models of "classification of types of transactions"

The data flow architecture for the solution is given below:


Raw data
prompt engineering
label generation using openAI
Manual testing of Generated label from OpenAI - 500 random label generation
Labeled dataset:
1.Gas
3.Food
4.Personal
5.Bank Business Transfer
6.Online purchase
Traditional NLP based classification algorithm using TFidfvectorizer
82% Accuracy
Use of Embeddings and PCA to develop Classification problem
85% Accuracy

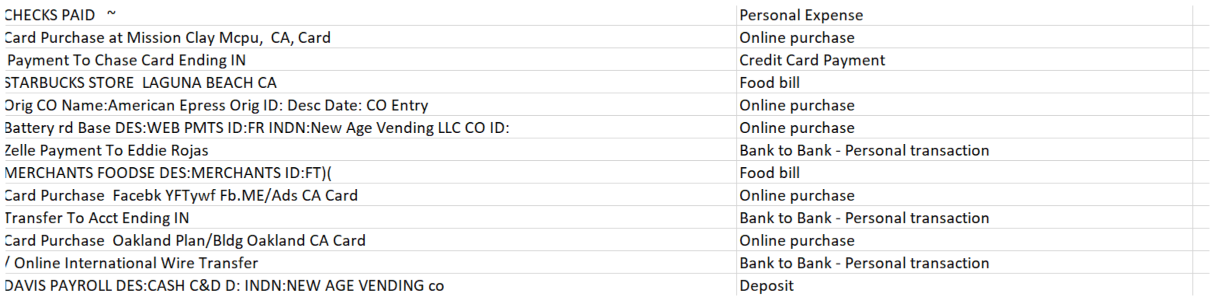
The primary goal here is to generate the labels first and then train a supervised model using these generated data to classify the types of transaction.

We have been given a non-labelled data of bank transactions, these data have various levels of transactions in it and it may happen that we will get completely new input data in future. The goal is to minimize or (completely get rid of expensive models like OpenAI to get the final prediction) and use comprehensive supervised classification model to predict the old as well as completely new kind of input data into its respective class

1. Process of Labelling the dataset:

we have almost 5000, samples (raw) after cleaning the provided input dataset.   These data samples are not labelled and its quite hard to tag them with label manually, here we have introduced the first block of the solution: **<u>use of OpenAI to label the raw dataset</u>**

As you can see that using prompts, we have passed the raw input (cleaned) data into the OpenAI LLM and asked them to given them the label as per its description. Below is the snapshot of the labels provided by openai for some of the complex description:



CHECKS PAID ~	Personal Expense
Card Purchase at Mission Clay Mcpu, CA, Card	Online purchase
Payment To Chase Card Ending IN	Credit Card Payment
STARBUCKS STORE LAGUNA BEACH CA	Food bill
Orig CO Name:American Epress Orig ID: Desc Date: CO Entry	Online purchase
Battery rd Base DES:WEB PMTS ID:FR INDN:New Age Vending LLC CO ID:	Online purchase
Zelle Payment To Eddie Rojas	Bank to Bank - Personal transaction
MERCHANTS FOODSE DES:MERCHANTS ID:FT)(	Food bill
Card Purchase Facebk YFTywf Fb.ME/Ads CA Card	Online purchase
Transfer To Acct Ending IN	Bank to Bank - Personal transaction
Card Purchase Oakland Plan/Bldg Oakland CA Card	Online purchase
/ Online International Wire Transfer	Bank to Bank - Personal transaction
DAVIS PAYROLL DES:CASH C&D D: INDN:NEW AGE VENDING co	Deposit

2.Validate the labels by OpenAI:

In this simple step, we manually picked random samples from the OpenAI and validated the labels, the temperature and Top-p hyperparameters were set to minimum and the prompts were optimally designed therefore we found that almost each description was labelled correctly

Below are the total labels we used:

1.Personal Expense: Education, Entertainment, Travel etc

2.Online Purchase: Online shopping, Subscriptions, mandates etc

3.Food Bill: Restaurant bill, Online Food Bill etc

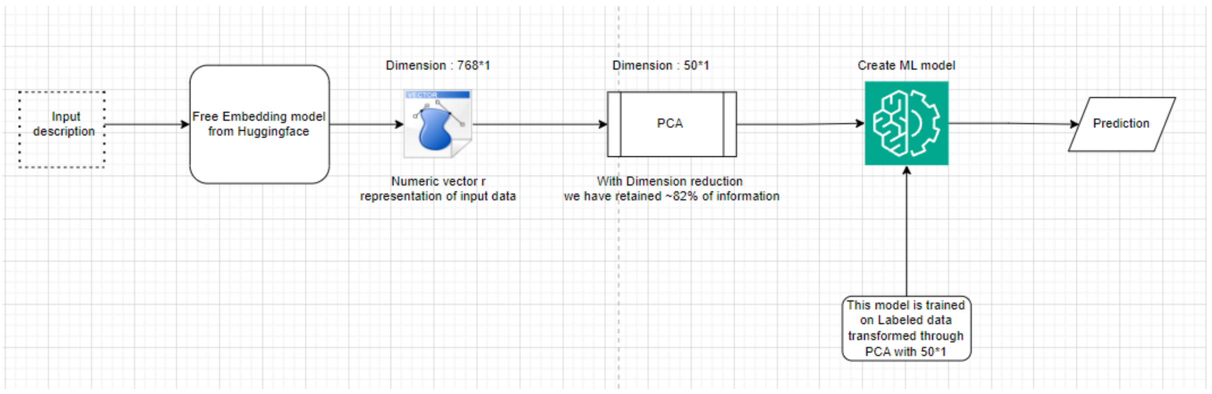
4. Card Payment: transaction where the details of "what" transaction are not defined but card details are mentioned

5. Bank to Bank /Deposits - Personal transaction: Transaction done to other banks or person

Once we have validated the transaction type on selected data samples, we are now ready to develop machine learning model

3.Model Generation:

Although I have designed 2 different ways to predict the type of transaction, the best way comes out as the 2nd way of "Use of Embeddings with PCA", so in this document I would be explaining the working of this model.


Input description
Free Embedding model from Huggingface
Dimension : 768*1
VECTOR
Numeric vector r representation of input data
Dimension : 50*1
PCA
With Dimension reduction we have retained ~82% of information
Create ML model
This model is trained on Labeled data transformed through PCA with 50*1
Prediction

Flowchart of "Embedding model with PCA."

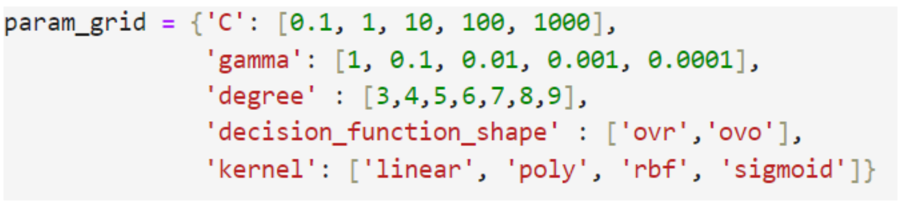
The input data is first converted into embedding vector (Numerical representation of the description) , This embedding or vector is of dimension 768*1 , to compress this size and reduce the dimension we used PCA , the usage of PCA was subjected to one constrain : "we should retain at least 80% of the 768 dimension's information"

The optimal number of Principal components which were retaining almost ~82% of original information was developed and we successfully converted the information from 768*1 Dimension to 50*1 dimension

We then used this 50*1 data to develop a ML model, below are the summary of ML model developed on this dataset

Note : All the models are developed using Gridsearch with analysis of different parameters like mean_test_score , mean_train_score to validate if the models are overfitting or underfitting , thus these developed models are reliable

1.SVC : Support vector Machine Classifier works very good on non-linear classification problem , given that the nature of data and high dimensions , this algorithm was natural first choice , the algorithm was build on following hyperparameters :

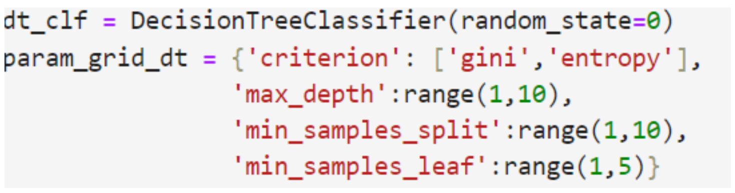


```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'degree' : [3,4,5,6,7,8,9],
              'decision_function_shape' : ['ovr','ovo'],
              'kernel': ['linear', 'poly', 'rbf', 'sigmoid']}
```

The results was: Accuracy of ~85%

2.Decision Tree: This rule-based classifier is again a good choice for the non-linear Multiclass classification problem :

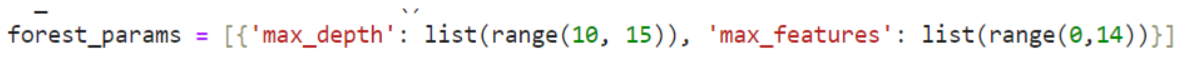
This algorithm was build on following hyperparameters :



```
dt_clf = DecisionTreeClassifier(random_state=0)
param_grid_dt = {'criterion': ['gini','entropy'],
                 'max_depth':range(1,10),
                 'min_samples_split':range(1,10),
                 'min_samples_leaf':range(1,5)}
```

The results were : Accuracy of ~ 71.2%

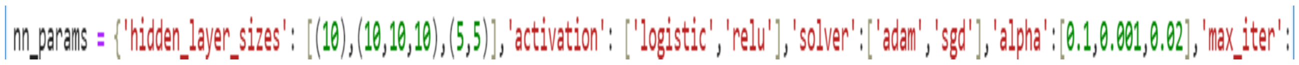
3. Randomforest classifier : This is a type of bagging model , although the size / number of samples of data are quite small , we tried this model with following hyperparameters :



```
forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range(0,14))}]
```

The results were : Accuracy of ~82%

4.Nueral Network (Multi layered Perceptron) : This Advancend Deep learning model was applied on following hyperparameters :



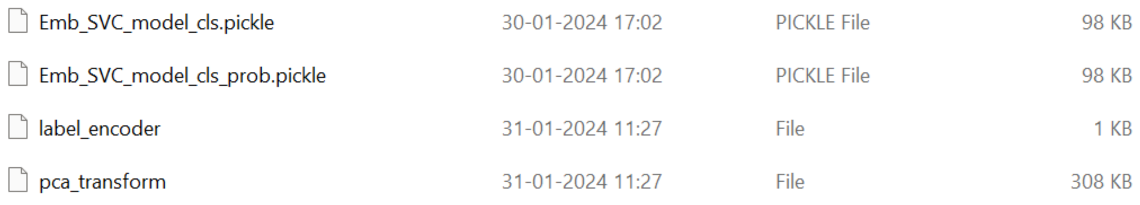
```
nn_params = {'hidden_layer_sizes': [(10),(10,10,10),(5,5)],'activation': ['logistic','relu'],'solver':['adam','sgd'],'alpha':[0.1,0.001,0.02],'max_iter':
```

(For better picture please refer to training notebook)

The results were: Accuracy of ~79.5%

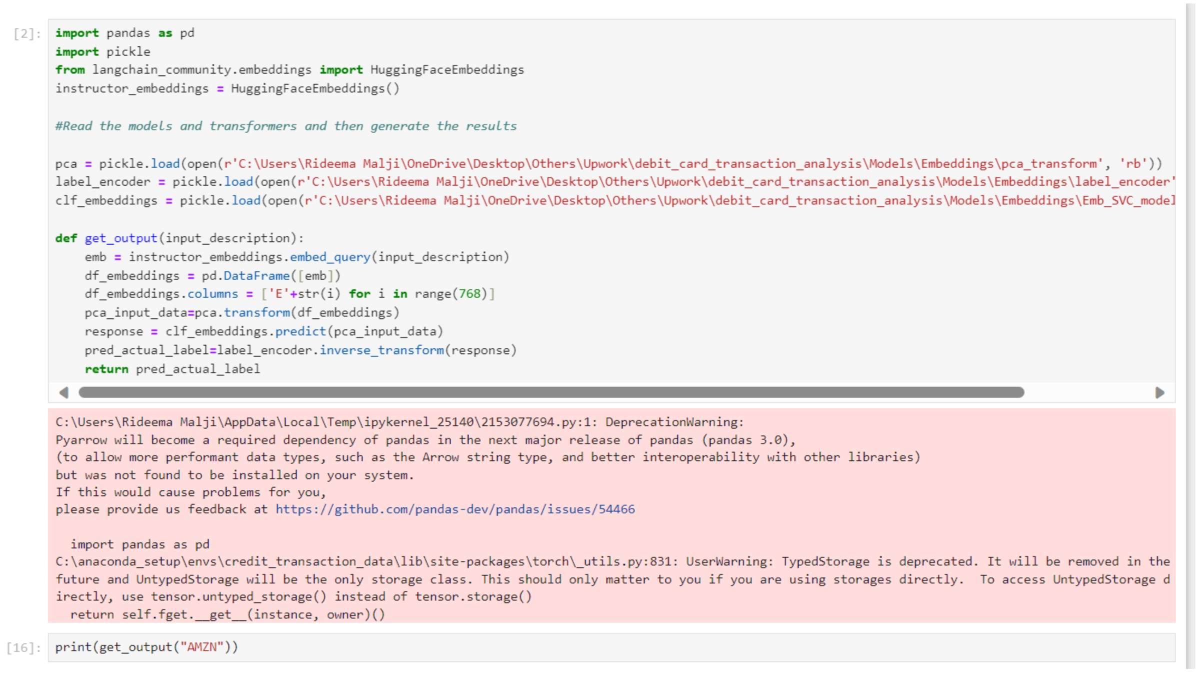
From the list of developed models, we finally selected the Support vector machine classifier model (SVC) for productionising:

To utilise these models, we saved all the preprocessing operations locally



Emb_SVC_model_cls.pickle	30-01-2024 17:02	PICKLE File	98 KB
Emb_SVC_model_cls_prob.pickle	30-01-2024 17:02	PICKLE File	98 KB
label_encoder	31-01-2024 11:27	File	1 KB
pca_transform	31-01-2024 11:27	File	308 KB

We can now use these stored models and preprocessing transformation whenever we want to use these models in the prediction:



```
[2]: import pandas as pd
     import pickle
     from langchain_community.embeddings import HuggingFaceEmbeddings
     instructor_embeddings = HuggingFaceEmbeddings()

     #Read the models and transformers and then generate the results

     pca = pickle.load(open(r'C:\Users\Rideema Malji\OneDrive\Desktop\Others\Upwork\debit_card_transaction_analysis\Models\Embeddings\pca_transform', 'rb'))
     label_encoder = pickle.load(open(r'C:\Users\Rideema Malji\OneDrive\Desktop\Others\Upwork\debit_card_transaction_analysis\Models\Embeddings\label_encoder'
     clf_embeddings = pickle.load(open(r'C:\Users\Rideema Malji\OneDrive\Desktop\Others\Upwork\debit_card_transaction_analysis\Models\Embeddings\Emb_SVC_model

     def get_output(input_description):
         emb = instructor_embeddings.embed_query(input_description)
         df_embeddings = pd.DataFrame([emb])
         df_embeddings.columns = ['E'+str(i) for i in range(768)]
         pca_input_data=pca.transform(df_embeddings)
         response = clf_embeddings.predict(pca_input_data)
         pred_actual_label=label_encoder.inverse_transform(response)
         return pred_actual_label
```

```
C:\Users\Rideema Malji\AppData\Local\Temp\ipykernel_25140\2153077694.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
C:\anaconda_setup\envs\credit_transaction_data\lib\site-packages\torch\_utils.py:831: UserWarning: TypedStorage is deprecated. It will be removed in the
future and UntypedStorage will be the only storage class. This should only matter to you if you are using storages directly.  To access UntypedStorage d
irectly, use tensor.untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
```

```
[16]: print(get_output("AMZN"))
```