CREDIT CARD FRAUD TRANSACTION PREDICTION

FINAL REPORT

Batch details	PGPDSE-FT Offline BLR June22
Team members	1. Medisetty Jagapathi Ramayya 2. Shoubir Kharghoria 3. Nandha Kumar 4. Neeharika Senapathi 5. Vangala Bharath
Domain of Project	Finance
Proposed project title	Fraud Detection in Credit Card Transaction
Group Number	Capstone Project Group 2
Team Leader	Neeharika Senapathi
Mentor Name	Mr.Animesh Tiwari

Date: 28-11-2022

Mr.Animesh Tiwari
Signature of the Mentor

Neeharika Senapati
Signature of the Team Leader

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Firstly, we would like to express our sincere gratitude to our mentor Mr.Animesh Tiwari for helping and guiding us throughout the entire duration of this project without which it would have been impossible to deliver such accurate results within the stipulated time period. Moreover, we would like to express our special thanks to Great Learning who gave us this golden opportunity to learn so many interesting concepts and apply them on such an important topic as a capstone project. This journey has made us more inquisitive about the never-ending possibilities with Machine learning and Data science in the near future and has helped us to successfully identify the path which we would like to pursue as my profession.

Last but not the least, we would like to extend our warm wishes and thanks to all our fellow team mates for their commitments, encouragements and continued support during the course of this project.

INDUSTRY REVIEW

Credit card fraud occurs mostly as a result of client carelessness, account takeover, skimming, Telephone phishing etc over which credit card companies have little control. However, credit card firms have a level of control over the transactions. With the increased use of online transactions and hacker assaults, gathering information and identifying patterns of fraudulent transactions has become critical in order to prevent such transactions from occurring and, as a result, to offer a sense of security to their clients.

Business problem statement (GOALS)

1. Business Problem Understanding There have always been those who would develop new ways to illegally access someone's funds. Due to the boom of technology, people are more sophisticated and reliable towards online transactions. Even though the result remains same. Since all transactions are through online with the simple entry of your credit card information. When a data breach results in monetary theft eventually, the loss of client loyalty as well as the company's image. organizations, consumers, banks, and merchants are all put at risk.

2. Business Objective Fraud detection is defined as "a series of efforts undertaken to prevent money or property from being gained via deception." To make a judgment, the majority of detection systems use a range of fraud detection datasets to generate a linked picture of both legitimate and invalid payment data. This choice must take into account IP address, Geo-location, device identity, "BIN" data, global latitude/longitude, historical transaction trends, and transaction information. In, this implies that merchants and issuers utilize analytically based solutions to identify fraud, which leverage internal and external data to apply a set of business rules or

analytical algorithms.

3. Approach Credit Card Fraud Detection using Machine Learning is a method that involves a Data Science team investigating data and developing a model that will deliver the best outcomes in detecting and preventing fraudulent transactions. This is accomplished by aggregating all relevant data of card users' transactions, such as the date, user zone, product category, amount, provider, client behaviour patterns, and so on. Exploration entails categorizing, aggregating, and segmenting data in order to scan millions of transactions for trends and detect fraud.

4. Conclusions

As demonstrated in this basic example, machine learning improves speed and accuracy in the battle against credit card fraud. Visa's AI/ML algorithms are predicted to have averted $25 billion in credit card fraud between 2010 and 2020. We can tackle such instances of theft by leveraging the capabilities of machine intelligence and deep learning. Fraud is a significant issue for the whole credit card business, and it is becoming more prevalent as electronic money transfers gain popularity. . Based on information about each card-holder's activity, machine learning-based solutions may continually enhance the accuracy of fraud protection.

DATASET AND DOMAIN

Data is collected from Kaggle.com.
Total number of Numerical columns - 10
Total number of Categorical columns - 12

DATA EXPLORATION (EDA)

DATATYPES INFO

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 555719 entries, 0 to 555718
Data columns (total 22 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   trans_date_trans_time  555719 non-null  object
 1   cc_num                 555719 non-null  int64
 2   merchant               555719 non-null  object
 3   category               555719 non-null  object
 4   amt                    555719 non-null  float64
 5   first                  555719 non-null  object
 6   last                   555719 non-null  object
 7   gender                 555719 non-null  object
 8   street                 555719 non-null  object
 9   city                   555719 non-null  object
 10  state                  555719 non-null  object
 11  zip                    555719 non-null  int64
 12  lat                    555719 non-null  float64
 13  long                   555719 non-null  float64
 14  city_pop               555719 non-null  int64
 15  job                    555719 non-null  object
 16  dob                    555719 non-null  object
 17  trans_num              555719 non-null  object
 18  unix_time              555719 non-null  int64
 19  merch_lat              555719 non-null  float64
 20  merch_long             555719 non-null  float64
 21  is_fraud               555719 non-null  int64
dtypes: float64(5), int64(5), object(12)
memory usage: 97.5+ MB
```

INFERENCE

- The dataset has 555718 records and 22 attributes.
- 5 features are of int, 5 float and 11 object datatype.
- It has 0 duplicated data points.

```
# For Numerrical columns
df.describe(include='number')
```

	cc_num	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long	is_fraud
count	5.557190e+05	555719.000000	555719.000000	555719.000000	555719.000000	5.557190e+05	5.557190e+05	555719.000000	555719.000000	555719.000000
mean	4.178387e+17	69.392810	48842.628015	38.543253	-90.231325	8.822189e+04	1.380679e+09	38.542798	-90.231380	0.003860
std	1.309837e+18	156.745941	26855.283328	5.061336	13.721780	3.003909e+05	5.201104e+06	5.095829	13.733071	0.062008
min	6.041621e+10	1.000000	1257.000000	20.027100	-165.672300	2.300000e+01	1.371817e+09	19.027422	-166.671575	0.000000
25%	1.800429e+14	9.630000	26292.000000	34.668900	-96.798000	7.410000e+02	1.376029e+09	34.755302	-96.905129	0.000000
50%	3.521417e+15	47.290000	48174.000000	39.371600	-87.476900	2.408000e+03	1.380762e+09	39.376593	-87.445204	0.000000
75%	4.635331e+15	83.010000	72011.000000	41.894800	-80.175200	1.968500e+04	1.385867e+09	41.954163	-80.264637	0.000000
max	4.992346e+18	22768.110000	99921.000000	65.689900	-67.950300	2.906700e+06	1.388534e+09	66.679297	-66.952026	1.000000

DESCRIPTION OF CATEGORICAL FEATURES

```python
# For Categorical Columns
df.describe(include='object').T
```

	count	unique	top	freq
merchant	555719	693	fraud_Kilback LLC	1859
category	555719	14	gas_transport	56370
first	555719	341	Christopher	11443
last	555719	471	Smith	12146
gender	555719	2	F	304886
street	555719	924	444 Robert Mews	1474
city	555719	849	Birmingham	2423
state	555719	50	TX	40393
job	555719	478	Film/video editor	4119
trans_num	555719	555719	2da90c7d74bd46a0caf3777415b3ebd3	1
week_day	555719	7	Monday	115136
month_name	555719	7	December	139538
age_group	555719	4	Senior citizen	294790

INFERENCE

- Merchant has 693 unique columns
- Category has 14 unique columns
- 50 unique states are there.

NULL VALUES

```
df.isnull().sum()
```

```
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

INFERENCE

.There are no null values in the dataset.

PROJECT JUSTIFICATION

PROJECT STATEMENT

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. This model is then used to identify whether a new transaction is fraudulent or not.

COMPLEXITY INVOLVED

- The target variable is imbalanced.
- Many redundant features present in the dataset.
- Collinearity amongst several features.
- Huge outliers present in the dataset.
- Since it is a high dimensional dataset, it will lead to high computational complexities.

EXPLORATORY DATA ANALYSIS (EDA)

CHECK FOR MULTICOLLINEARITY (VIF)

	VIF_Factor	Features
3	570.956798	long
7	565.465029	merch_long
5	168.537981	unix_time
2	77.959500	lat
6	77.852292	merch_lat
1	6.411975	zip
4	1.027424	city_pop
0	1.000017	amt

INFERENCE

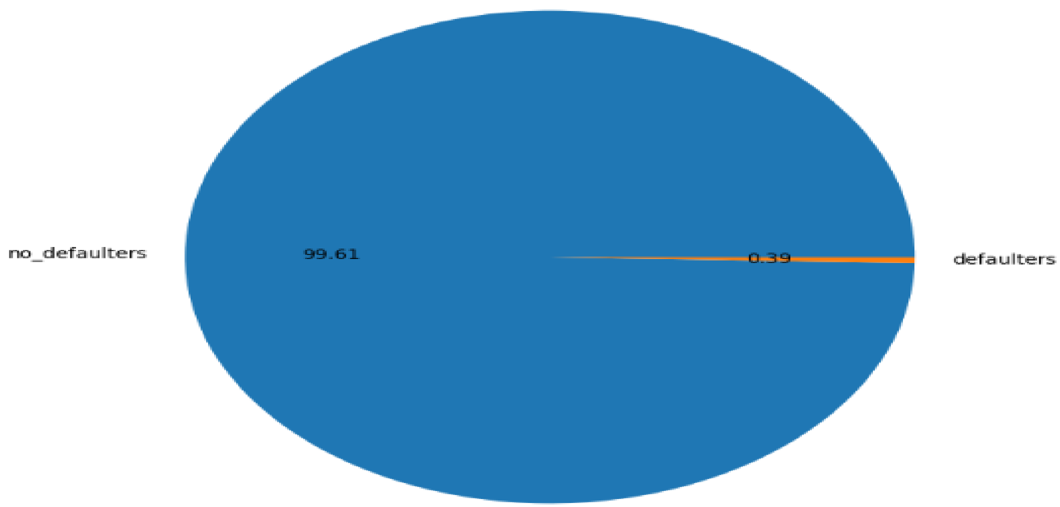
VIF is very high for some featues , so we decided not to use all the features to build our basic model and we shall build the upcoming models based on the p_value of each feature we'll decide it's significance for our prediction. If we find it insignificant, we’ll drop it.

DISTRIBUTION OF VARIABLES

DISTRIBUTION OF TRAGET COLUMN

```
plt.figure(figsize=[15,8])
plt.pie(m, labels=['no_defaulters', 'defaulters'], autopct='%.2f')

([<matplotlib.patches.Wedge at 0x17e85d57310>,
  <matplotlib.patches.Wedge at 0x17e85d57a00>],
 [Text(-1.0999191268840707, 0.013338452480854692, 'no_defaulters'),
  Text(1.0999191263279473, -0.013338498339964886, 'defaulters')],
 [Text(-0.5999558873913112, 0.007275519535011649, '99.61'),
  Text(0.5999558870879712, -0.007275544549071741, '0.39')])
```



99.61% people are non-fraud, 0.39% are fraud in the target variable 'is-fraud'

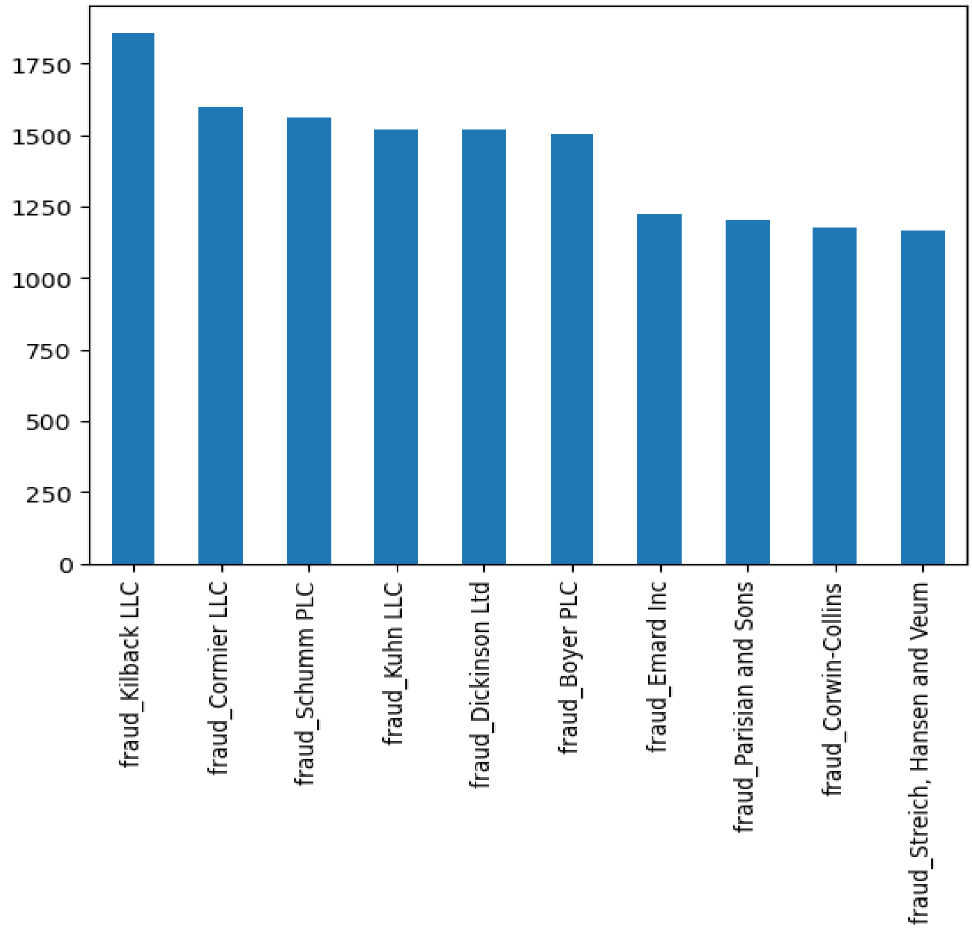
CLASS IMBALANCE AND ITS TREATMENT

- The target column is imbalanced and this can lead to bias during prediction if we go ahead building a model with this data set.
- Initially, we have gone ahead with building our model without treatment.
- In the further model's built, we shall be using SMOTE Techniques to overcome this Class Imbalance in the target column.

Analysis of Categorical Features.

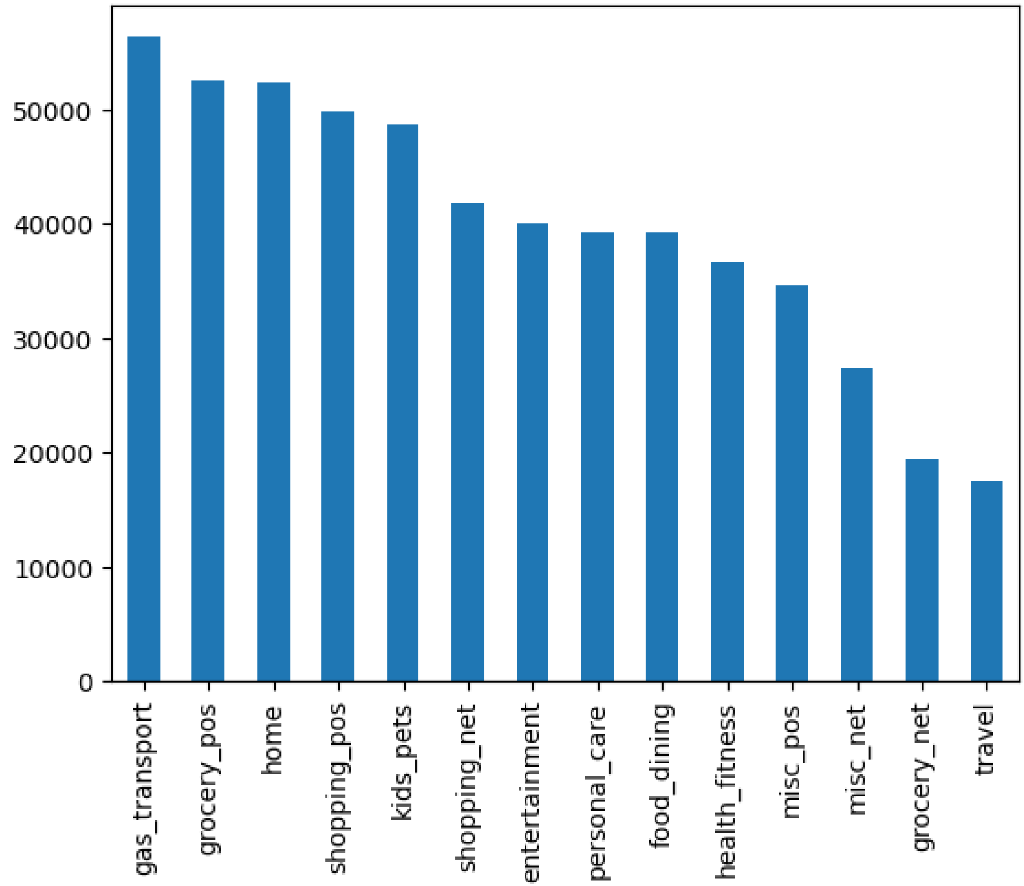
```
df_cat['merchant'].value_counts().head(10).plot(kind='bar')
```

<AxesSubplot:>



```
df_cat['category'].value_counts().plot(kind='bar')
```

<AxesSubplot:>



INFERENCE

- Most credit card transactions are happeining in gas_tansport
- The second fallowed by grocery pos
- The top merchants are killback llc
- The second top merchant is cormier llc

PRESENCE OF OUTLIERS AND ITS TREATMENT

INFERENCE

Since this is a credit card transaction data we are keeping the outliers .

STATES WITH MOST FRAUD TRANSACTIONS

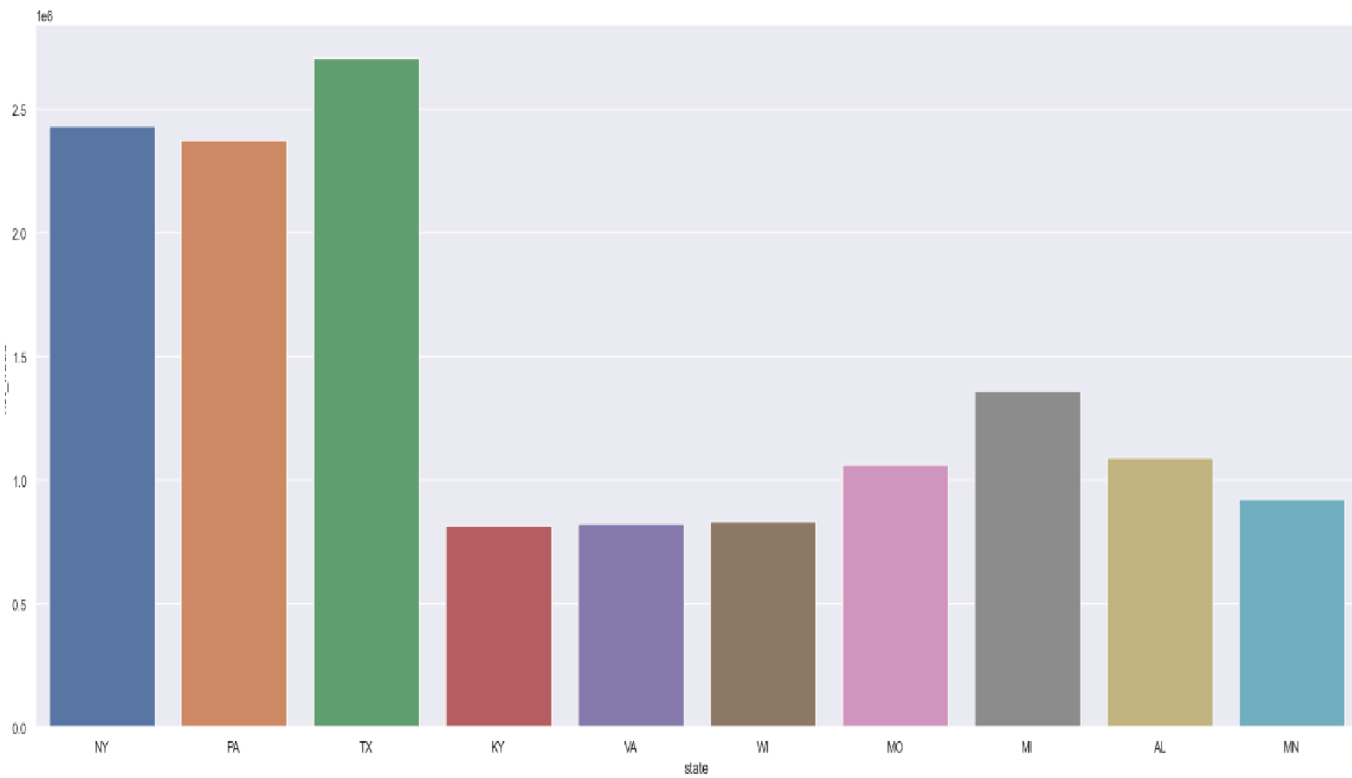
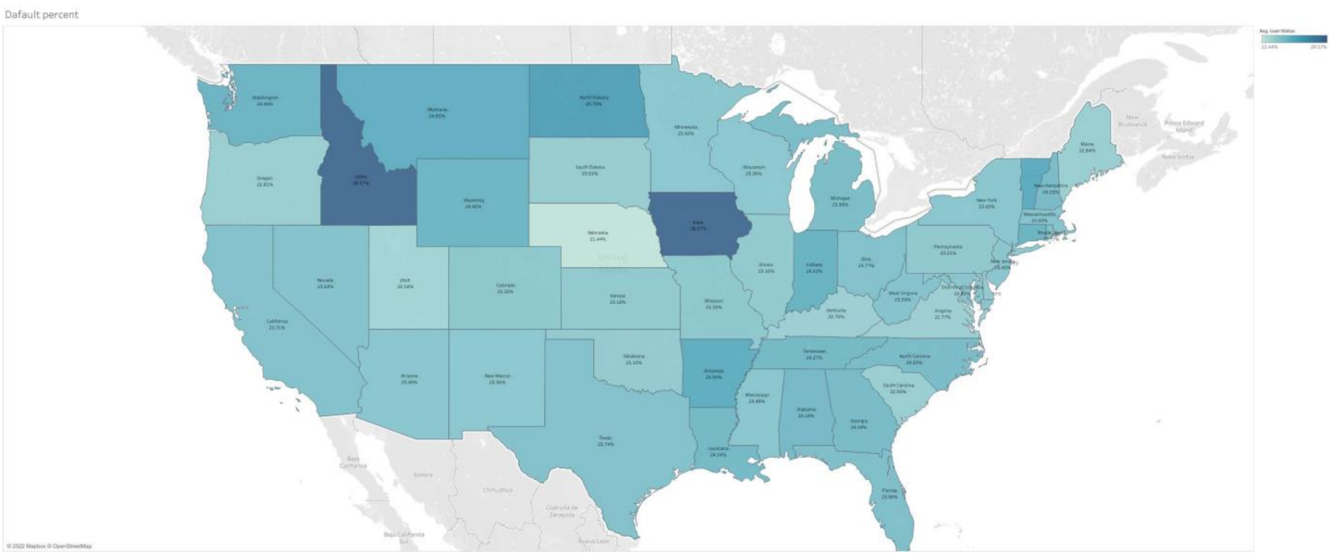


TABLEAU REPRESENTATIONS:

Demographic map of percent of loan defaulters across the states of USA



INFERENCE

These are all the states of USA in which more fraud transactions are happening in new york , phillidelphia and texas.

STATISTICAL SIGNIFICANCE

MANN-WHITNEY U TEST (for Numerical Features)

A Mann-Whitney U test (sometimes called the **Wilcoxon rank-sum test) is used to compare the differences between two independent samples** when the sample distributions are not normally distributed and the sample sizes are small (n <30). It is considered to be the nonparametric equivalent to the two-sample independent t-test. The hypotheses for the test are:

- H_0=the distribution of both samples are same/ both samples are independent
- H_1= the distribution of both samples are not same/ dependent

- Assuming the level of confidence as 95%.

```
#performing manvitneyu test
#Null hypothesis= the distribution of both samples are same/ both samples are independent
#Alternate hypothesis= the distribution of both samples are not same/ dependent
#assuming significance level is 5%
nonfraud=df[df['is_fraud']==0]
fraud= df[df['is_fraud']==1]
man_vit_p=[round(stats.mannwhitneyu(nonfraud[i],fraud[i])[1],3) for i in num.columns ]
```

```
manvit=pd.DataFrame({'Vaiable':num.columns,'P_value':man_vit_p})
manvit
```

	Vaiable	P_value
0	amt	0.000
1	zip	0.054
2	lat	0.000
3	long	0.820
4	city_pop	0.005
5	unix_time	0.000
6	merch_lat	0.000
7	merch_long	0.753
8	is_fraud	0.000
9	weekday_no	0.000
10	week_no	0.000
11	day_no	0.000

INFERENCE :

According to Mann and Whitney's U-test
Merchn_long , long are not effecting the target

CHI-SQUARE TEST OF INDEPENDENCE (for Categorical Features)

The Chi-Square Test of Independence determines whether there is an association between categorical variables (i.e., whether the variables are independent or related). It is a nonparametric test. The hypotheses for the test are

- H0: The variables are independent.
- H1: The variables are not independent (i.e., variables are dependent).

Assuming the level of confidence as 95%.

Job vs is_fraud(target)

```
#null: job and is_fraud are independent
#alter:job and is_fraud are dependent
```

```
df.job.nunique()
```

```
478
```

```
stat, p, dof, expected=stats.chi2_contingency(pd.crosstab(df_new['job'],df_new['is_fraud']))
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 0.0
Dependent (reject H0)
```

#pvalue is less than significance level we reject null hypothesis concluding that job and is_fraud are dependent

Age_group vs is_fraud(target)

```
#null: age_group and is_fraud are independent
#alter:age_group and is_fraud are dependent
```

```
df.age_group.nunique()
```

```
4
```

```
stat, p, dof, expected=stats.chi2_contingency(pd.crosstab(df_new['age_group'],df_new['is_fraud']))
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 1.1255254763105812e-05
Dependent (reject H0)
```

#pvalue is less than significance level we reject null hypothesis concluding that age_group and is_fraud are dependent

INFERENCE:

We tested if every categorical feature is dependent on the Target Variable, the p-value of all the features came out to be less than 0.05 (Level of Significance) and thus we reject Null hypothesis. Therefore, all the categorical features are dependent on the Target Variable.

FEATURE ENGINEERING

NUMERICAL FEATURES SCALING

We scale the variables to get all the variables in the same range. With this, we can avoid a problem in which some features come to dominate solely because they tend to have larger values than others.

scaling the data

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train= pd.DataFrame(ss.fit_transform(X_train),columns=X_train.columns)
X_test = pd.DataFrame(ss.transform(X_test) , columns = X_test.columns)
```

```
X_train.head()
```

	amt	lat	long	city_pop	merch_lat	merch_long	weekday_no	week_no	day_no	min_day	...	week_day_Wednesday	month_name_Dece
0	-0.162681	0.005030	-1.405948	-0.260159	-0.086688	-1.384557	-1.250118	0.916536	0.730005	-0.892846	...	-0.324063	-0.5
1	1.504974	-0.158237	0.712742	-0.277399	-0.269878	0.677487	0.585953	-0.129111	0.953382	0.492007	...	-0.324063	-0.5
2	-0.146571	1.090279	1.477933	-0.281881	1.089653	1.525750	1.503988	-0.593844	1.511824	-0.027313	...	-0.324063	-0.5
3	-0.141418	1.199243	1.112898	-0.292347	1.368482	1.092216	-0.791100	-1.523308	1.511824	0.088091	...	-0.324063	-0.5
4	0.027636	0.697697	1.109837	-0.288503	0.827555	1.113211	1.044970	1.032719	-1.280388	-0.604335	...	-0.324063	1.7

5 rows × 521 columns

ENCODING OF CATEGORICAL FEATURES

We have used pandas get dummies to encode the categorical features

```
#after enconding
encoding=pd.get_dummies(df_categorical,drop_first=True)
encoding.head()
```

	category_food_dining	category_gas_transport	category_grocery_net	category_grocery_pos	category_health_fitness	category_home	category_kids_pets	categ
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
2	0	0	0	0	1	0	0	
3	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	

5 rows × 506 columns

```
## we will conat the numerical and categorical variables
```

BASE MODEL (LOGISTIC REGRESSION)

BUILD A FULL LOGISTIC MODEL ON A TRAINING DATASET

```
log_reg.summary()
```

Logit Regression Results

Dep. Variable:	is_fraud	**No. Observations:**	389003
Model:	Logit	**Df Residuals:**	388975
Method:	MLE	**Df Model:**	27
Date:	Tue, 29 Nov 2022	**Pseudo R-squ.:**	0.2934
Time:	11:22:24	**Log-Likelihood:**	-6968.3
converged:	True	**LL-Null:**	-9862.1
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	149.9784	21.012	7.138	0.000	108.796	191.160
amt	1.5708	0.035	44.651	0.000	1.502	1.640
lat	0.2551	0.240	1.063	0.288	-0.215	0.725
city_pop	0.0084	0.030	0.283	0.777	-0.050	0.066
merch_lat	-0.1609	0.240	-0.670	0.503	-0.632	0.310
year_dayno	33.1150	4.733	6.996	0.000	23.838	42.392
no_of_years	0.3371	0.058	5.846	0.000	0.224	0.450
weekday_no	-0.4079	0.168	-2.422	0.015	-0.738	-0.078
month	-2.5768	3.403	-0.757	0.449	-9.247	4.094
week_no	-3.2926	1.171	-2.812	0.005	-5.588	-0.997
day_no	-0.1012	0.112	-0.907	0.364	-0.320	0.117
hr_day	0.1055	0.005	19.337	0.000	0.095	0.116

hr_day	0.1055	0.005	19.337	0.000	0.095	0.116
category_food_dining	0.2621	0.223	1.173	0.241	-0.176	0.700
category_gas_transport	1.8552	0.195	9.497	0.000	1.472	2.238
category_grocery_net	1.9305	0.254	7.598	0.000	1.432	2.428
category_grocery_pos	2.2968	0.176	13.070	0.000	1.952	2.641
category_health_fitness	0.1025	0.228	0.450	0.653	-0.344	0.549
category_home	-0.1086	0.215	-0.506	0.613	-0.530	0.312
category_kids_pets	-0.0243	0.213	-0.114	0.909	-0.442	0.393
category_misc_net	1.8015	0.185	9.718	0.000	1.438	2.165
category_misc_pos	0.3993	0.219	1.820	0.069	-0.031	0.829
category_personal_care	0.4628	0.209	2.210	0.027	0.052	0.873
category_shopping_net	1.3509	0.178	7.594	0.000	1.002	1.700
category_shopping_pos	0.3446	0.190	1.812	0.070	-0.028	0.717
category_travel	-1.7872	0.293	-6.090	0.000	-2.362	-1.212
age_group_Senior citizen	-0.0359	0.099	-0.362	0.717	-0.230	0.158
age_group_Very young age	0.3902	0.150	2.609	0.009	0.097	0.683
age_group_Young age	0.1967	0.100	1.970	0.049	0.001	0.392

Possibly complete quasi-separation: A fraction 0.17 of observations can be perfectly predicted. This might indicate that there is complete quasi-separation. In this case some parameters will not be identified.

INFERENCE

- LLR p-value of the model is 0.000, which is less than 0.05 therefore Null hypothesis is rejected and thus, there is at least one feature which is significant.
- Pseudo R-square of the model is: 0.2934 , it's far from 1 therefore we conclude that there are many improvements to be done on the model.
- Log-Likelihood of the model is: -6968.3 which is greater than the Log-Likelihood of the Null Model i.e., -9862. Indicating our model has performed quite better.

CLASSIFICATIONREPORT

```
Classification report train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    370891
           1       0.60      0.12      0.20      1440

    accuracy                           1.00    372331
   macro avg       0.80      0.56      0.60    372331
weighted avg       1.00      1.00      1.00    372331

Classification report test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    182683
           1       0.55      0.10      0.18       705

    accuracy                           1.00    183388
   macro avg       0.77      0.55      0.59    183388
weighted avg       0.99      1.00      0.99    183388
```

INFERENCE

- Since, we wouldn't want to wrongly classify the actual transaction as fraud i.e., reduce Type-II Error as much as possible.
- This can be done by focussing on the recall i.e., 0.12 for the model, since it is a component of Type-II Error which effects the prediction of the model.

DECISION TREE

CLASSIFICATION REPORT

```
Classification report train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    387498
           1       0.99      0.88      0.94      1505

    accuracy                           1.00    389003
   macro avg       1.00      0.94      0.97    389003
weighted avg       1.00      1.00      1.00    389003

Classification report test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.81      0.75      0.78       640

    accuracy                           1.00    166716
   macro avg       0.91      0.87      0.89    166716
weighted avg       1.00      1.00      1.00    166716
```

INFERENCE

Decision Tree is a non-parametric supervised learning method. It builds a model in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets, which is called splitting.

The 0 class has been predicted correctly because we have 99 percent of

0 class .The recall score of the 0.88 in the train dataset and 0.78 in the

Test dataset

RANDOM FOREST

CLASSIFICATION REPORT

```
0.9986144101346002
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    387498
           1       1.00      1.00      1.00      1505

    accuracy                           1.00    389003
   macro avg       1.00      1.00      1.00    389003
weighted avg       1.00      1.00      1.00    389003

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.97      0.66      0.78       640

    accuracy                           1.00    166716
   macro avg       0.99      0.83      0.89    166716
weighted avg       1.00      1.00      1.00    166716
```

INFERNECE

.The Random forest model is overftitting in the train and test

.So we do hyperparamter tuning and check the results.

RANDOM FOREST AFTER TUNING

CLASSIFICATION REPORT

```
: rfc = RandomForestClassifier(criterion='entropy', min_samples_leaf=5, min_samples_split= 5, n_estimators=20 )
rfc.fit(X_train , y_train)
y_train_pred_rfc=rfc.predict(X_train)
y_test_pred_rfc=rfc.predict(X_test)
print(accuracy_score(y_test , y_test_pred_rfc))
print(classification_report(y_train, y_train_pred_rfc))
print(classification_report(y_test  , y_test_pred_rfc))
```

```
0.9985424314402936
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    387498
           1       0.99      0.70      0.82      1505

    accuracy                           1.00    389003
   macro avg       1.00      0.85      0.91    389003
weighted avg       1.00      1.00      1.00    389003

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.97      0.64      0.77       640

    accuracy                           1.00    166716
   macro avg       0.99      0.82      0.88    166716
weighted avg       1.00      1.00      1.00    166716
```

INFERNECE

.After tuning the model with grid search cv , we got the best params as Criterion as entropy , min_sample_leaf = 5 , min_sample_split =5 , n-estimators = 20

.The class 0 is predicted with 100 percent accuracy and class 1 is predicted with 0.70

Training and 0.64 in the testing datset

BOOSTING MODELS

ADA BOOST

CLASSIFICATION REPORT

```
0.9969888912881787
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    387498
           1       0.67      0.35      0.46      1505

    accuracy                           1.00    389003
   macro avg       0.83      0.67      0.73    389003
weighted avg       1.00      1.00      1.00    389003

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.69      0.39      0.50       640

    accuracy                           1.00    166716
   macro avg       0.85      0.69      0.75    166716
weighted avg       1.00      1.00      1.00    166716

training accuracy for ada50  0.9968123639149312
testing accuracy for ada50 0.9969888912881787
```

INFERENCE

The ada boost recall score on the training dataset is 0.35 for 1 class
The ada boost recall score on the testing data set is 0.39
We will do the hyperparamter tu

ADA BOOST AFTER HYPERPARAMETER TUNING

CLASSIFICATION REPORT

```
0.9968989179202956
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    387498
           1       0.64      0.34      0.45      1505

    accuracy                           1.00    389003
   macro avg       0.82      0.67      0.72    389003
weighted avg       1.00      1.00      1.00    389003

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.67      0.37      0.48       640

    accuracy                           1.00    166716
   macro avg       0.84      0.69      0.74    166716
weighted avg       1.00      1.00      1.00    166716

training accuracy for ada50  0.9966992542473966
testing accuracy for ada50 0.9968989179202956
```

INFERENCES

.After hyperparameter tuning the results of model has not been increased
Due to the reason the that our data set is highly imbalanced we will build
Other boosting models and check the results

BOOSTING MODELS

Gradient BOOST

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	387498
1	0.99	0.23	0.37	1505
accuracy			1.00	389003
macro avg	0.99	0.61	0.68	389003
weighted avg	1.00	1.00	1.00	389003

	precision	recall	f1-score	support
0	1.00	1.00	1.00	166076
1	0.98	0.20	0.33	640
accuracy			1.00	166716
macro avg	0.99	0.60	0.66	166716
weighted avg	1.00	1.00	1.00	166716

INFERENCES

. The gradient boosting model has predicted the 1 class correctly but The 0 class recall was very less

.This is due to the imbalanced in the dataset

XTREME GRADIENT BOOST

HYPER PARAMETER TUNING USING GRID SEARCH

Hyperparameters are the parameters in the model that are pre-set by the user. Grid Search considers all the combinations of hyperparameters and returns the best hyperparameter values. We pass some of the hyperparameters in the decision tree to the GridSearchCV () and build the tree using the optimal values obtained using GridSearch method.

```
##Tuning the paramters using Gridsearchcv , kfold
from  sklearn.model_selection import GridSearchCV , KFold

params = {'n_estimators' : [4,5,6,7,8,9,10] ,
          'learning_rate':[0.3, 0.31, 0.33, 0.34, 0.35],
          "min_child_weight" : [0,1] ,
          "max_depth" : [2,4,6,8,10] }
xg = XGBClassifier(random_state = 10)
xgcv  = GridSearchCV(xg, params , cv = 5, scoring  = "accuracy")
xgcv.fit(X_train , y_train)
xgcv.best_params_
```

```
{'learning_rate': 0.33,
 'max_depth': 8,
 'min_child_weight': 0,
 'n_estimators': 10}
```

XGB MODEL BUILT USING THE BEST PARAMETERS FROM GD

CLASSIFICATION REPORT

```
Classification report train
click to expand output; double click to hide output     support

           0       1.00      1.00      1.00    387498
           1       0.99      0.78      0.87      1505

    accuracy                           1.00    389003
   macro avg       0.99      0.89      0.93    389003
weighted avg       1.00      1.00      1.00    389003

Classification report test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    166076
           1       0.92      0.74      0.82       640

    accuracy                           1.00    166716
   macro avg       0.96      0.87      0.91    166716
weighted avg       1.00      1.00      1.00    166716
```

INFERENCE

- As XGBoost is the best performing model among all models built, we use that and further use GridSearchCV to tune the hyper parameters.
- And since it's computationally intensive and time consuming, we give a smaller number of inputs for tuning.
- Out of the given parameters, Learning Rate of 0.33 , min_child_weight = 0 , N_estimators = 10 , max depth of 8 have the best scores over all.
- With tuned parameters both Train and Test metrics have improved slightly.
- So far, this model gives the best scores.

FINAL INFERENCE

- In this project we made 5 algorithms. We started with the base model and took the inferences from them after evaluating important measures we went for their parameter tuning to increase their performance.
- Based on our objective, we had to focus on reducing the False Negatives i.e., the Type 2 Error, because predicting those who are actually fraud as non-fraud who cause the bank a major loss.
- Since, our data is highly imbalanced we mainly focus on the F1-Score, because the performance metrics 'accuracy' would be affected due to bias.
- As we can see we are getting best recall score and F1-Score in XGBoost model built with the significant features and grid search cv .