<u>Aim</u>: Experiment to build Business Intelligence Mini Project.

<u>To Do:</u>

a) Problem definition, identifying which data mining task is needed
b) Identify and use a standard data mining dataset available for the problem. Some links for data mining datasets are: WEKA, Kaggle, KDD cup, Data Mining Cup, UCI Machine Learning Repository etc.
c) Perform EDA and Visualization of the data available. Showcase the results using appropriate graphs.
d) Perform Data Preprocessing and Showcase the effect of it on original data.
e) Apply appropriate 2 algorithms(techniques) for given case study.
Build appropriate models for testing and training data.
Interpret all the model outputs and check the performance of all these models that you have built (test and train). Interpret and visualize the results
f)Use all the model performance measures you have
I have learned so far. Share your remarks on which model performs the best. Provide clearly the BI decision that is to be taken as a result of mining

<u>FRAUD DETECTION</u>

<u>Problem Definition</u>

Fraud detection is a common problem in various industries, including finance, insurance, and e-commerce. The problem involves identifying fraudulent transactions or activities among a large number of legitimate ones. Fraud can take many forms, including credit card fraud, insurance fraud, identity theft, and money laundering.

The goal of fraud detection is to build a system that can accurately distinguish between legitimate and fraudulent transactions or activities. The system typically uses historical data to learn patterns and detect anomalies that are indicative of fraud. The system should also be able to adapt to new types of fraud as they emerge.

Fraud detection is an important problem because it can help prevent financial losses, protect customer information, and maintain the integrity of a company's operations. A successful fraud detection system can also improve customer trust and confidence in the company's services.

Data Mining Task used:

Detection: One of the most common data mining tasks used in fraud detection is detection. Anomaly detection involves identifying transactions or activities that deviate significantly from normal patterns. In fraud detection, anomalies may indicate fraudulent behavior that requires further investigation.

Classification: Classification is another data mining task that is commonly used in fraud detection. In this task, a machine learning algorithm is trained to classify transactions or activities as either fraudulent or legitimate based on historical data.

About Dataset:

Link: https://www.kaggle.com/datasets/dermisfit/fraud-transactions-dataset

This dataset is fictional and is trying to simulate real life details. Any similarity to real life cases is purely coincidental.
It has the following columns.

trans_date_trans_time: The date and time of the transaction.
cc_num: credit card number.
merchant: Merchant who was getting paid.
category: In what area does that merchant deal.
amt: Amount of money in American Dollars.
first: first name of the card holder.
last: last name of the card holder.
gender: Gender of the cardholder.Just male and female!
street:Street of card holder residence
city:city of card holder residence
state:state of card holder residence , zip:ZIP code of card holder residence
lat:latitude of card holder,long:longitude of card holder
city_pop:Population of the city , job:trade of the card holder
dob:Date of birth of the card holder , trans_num: Transaction ID
unix_time: Unix time which is the time calculated since 1970 to today.
merch_lat: latitude of the merchant
merch_long:longitude of the merchant
is_fraud: Whether the transaction is fraud(1) or not(0)

Exploratory Data Analysis

- The info() method prints information about the DataFrame.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             555719 non-null  int64
 1   trans_date_trans_time  555719 non-null  object
 2   cc_num                 555719 non-null  int64
 3   merchant               555719 non-null  object
 4   category               555719 non-null  object
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object
 7   last                   555719 non-null  object
 8   gender                 555719 non-null  object
 9   street                 555719 non-null  object
 10  city                   555719 non-null  object
 11  state                  555719 non-null  object
 12  zip                    555719 non-null  int64
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64
 16  job                    555719 non-null  object
 17  dob                    555719 non-null  object
 18  trans_num              555719 non-null  object
 19  unix_time              555719 non-null  int64
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 97.5+ MB
```

- This describes the data types of the particular column

```
print(data.dtypes)
```

```
Unnamed: 0                 int64
trans_date_trans_time     object
cc_num                     int64
merchant                  object
category                  object
amt                      float64
first                     object
last                      object
gender                    object
street                    object
city                      object
state                     object
zip                        int64
lat                      float64
long                     float64
city_pop                   int64
job                       object
dob                       object
trans_num                 object
unix_time                  int64
merch_lat                float64
merch_long               float64
is_fraud                   int64
dtype: object
```

- The following implementation returns a description of the data in the DataFrame.

```
data.describe()
```

	Unnamed: 0	cc_num	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long	is_fraud
count	555719.000000	5.557190e+05	555719.000000	555719.000000	555719.000000	555719.000000	5.557190e+05	5.557190e+05	555719.000000	555719.000000	555719.000000
mean	277859.000000	4.178387e+17	69.392810	48842.628015	38.543253	-90.231325	8.822189e+04	1.380679e+09	38.542798	-90.231380	0.003860
std	160422.401459	1.309837e+18	156.745941	26855.283328	5.061336	13.721780	3.003909e+05	5.201104e+06	5.095829	13.733071	0.062008
min	0.000000	6.041621e+10	1.000000	1257.000000	20.027100	-165.672300	2.300000e+01	1.371817e+09	19.027422	-166.671575	0.000000
25%	138929.500000	1.800429e+14	9.630000	26292.000000	34.668900	-96.798000	7.410000e+02	1.376029e+09	34.755302	-96.905129	0.000000
50%	277859.000000	3.521417e+15	47.290000	48174.000000	39.371600	-87.476900	2.408000e+03	1.380762e+09	39.376593	-87.445204	0.000000
75%	416788.500000	4.635331e+15	83.010000	72011.000000	41.894800	-80.175200	1.968500e+04	1.385867e+09	41.954163	-80.264637	0.000000
max	555718.000000	4.992346e+18	22768.110000	99921.000000	65.689900	-67.950300	2.906700e+06	1.388534e+09	66.679297	-66.952026	1.000000

- This will print the total number of duplicate records present in the dataset.

```
[7] #Find the duplicates

    data.duplicated().sum()
```

```
0
```

- This will print the unique values present in that particular column.

```
[8] #unique values

    data['is_fraud'].unique()
```

```
array([0, 1])
```

- The following code will print the fraud happened as we gave the condition is_fraud = = 1.

```
data[data['is_fraud']==1].head()
```

ant	category	amt	first	last	gender	street	...	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_long	is_f
nill- ore	health_fitness	24.84	Brooke	Smith	F	63542 Luna Brook Apt. 012	...	31.8599	-102.7413	23	Cytogeneticist	1969-09-15	16bf2e46c54369a8eab2214649506425	1371852399	32.575873	-102.604290	
uez, kins	misc_net	780.52	Douglas	Willis	M	619 Jeremy Garden Apt. 681	...	42.5545	-90.3508	1306	Public relations officer	1958-09-10	ab4b379d2c0c9c667d46508d4e126d72	1371853942	42.461127	-91.147148	
now PLC	entertainment	620.33	Douglas	Willis	M	619 Jeremy Garden Apt. 681	...	42.5545	-90.3508	1306	Public relations officer	1958-09-10	47a9987ae81d99f7832a54b29a77bf4b	1371854247	42.771834	-90.158365	
ote, and	shopping_net	1077.69	William	Perry	M	458 Phillips Island	...	30.4590	-90.9027	71335	Herbalist	1994-05-31	fe956c7e4a253c437c18918bf96f7b62	1371854335	31.204974	-90.261595	

- The following code will print Standardized features by scaling each feature to a given range.

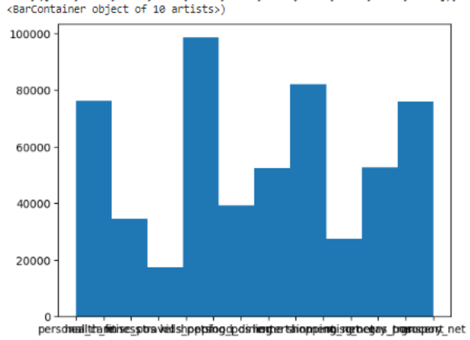
```
[14] # Scaling numerical variables using MinMaxScaler
     from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     data[['amt', 'city_pop']] = scaler.fit_transform(data[['amt', 'city_pop']])
```

Visualization

- The following graph shows the count of different categories.

```
# Creating a histogram showing count of each category
import matplotlib.pyplot as plt
plt.hist(data['category'])
```

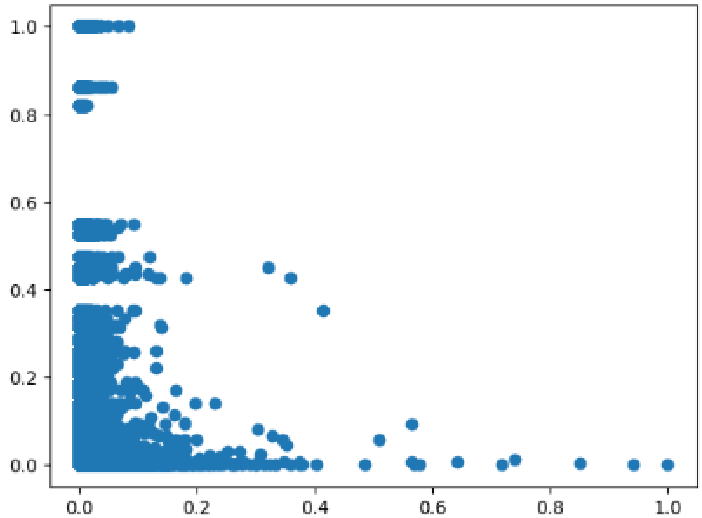
```
(array([76001., 34574., 17449., 98483., 39268., 52345., 81883., 27367.,
        52553., 75796.]),
 array([ 0. ,  1.3,  2.6,  3.9,  5.2,  6.5,  7.8,  9.1, 10.4, 11.7, 13. ]),
 <BarContainer object of 10 artists>)
```



- The following scatter plot graph shows the amount given by each city population

```
# Creating a scatter plot of amount given by each city population
plt.scatter(data['amt'], data['city_pop'])
```

```
<matplotlib.collections.PathCollection at 0x7f3ef058bf70>
```



- The pie chart shows Categories divided based on the amount given for it in percentage.

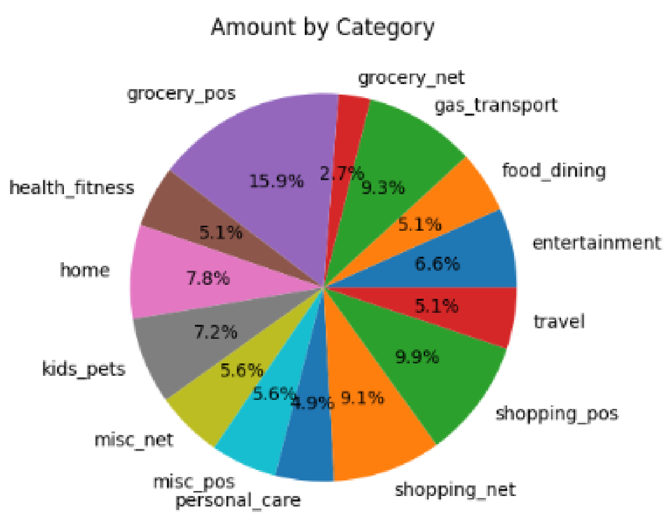
```
import pandas as pd
import matplotlib.pyplot as plt


# Group data by category
grouped_data = data.groupby(['category']).sum()

# Create pie chart
plt.pie(grouped_data['amt'], labels=grouped_data.index, autopct='%1.1f%%')

# Add title
plt.title('Amount by Category')

# Show chart
plt.show()
```



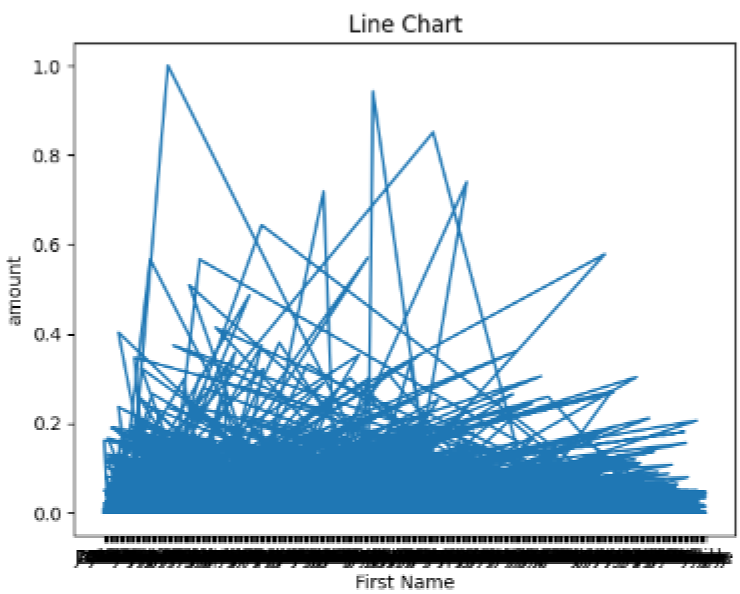
- The Line chart describes how much an amount a person has paid. The person's first name is taken into consideration.

```
import pandas as pd
import matplotlib.pyplot as plt

# Create a line chart
plt.plot(data['first'], data['amt'])

# Add x and y labels and a title
plt.xlabel('First Name')
plt.ylabel('amount')
plt.title('Line Chart')

# Display the chart
plt.show()
```



Data Preprocessing

- Taking care of the missing data by filling it with 0

```
[33] data.fillna(0)
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	lat	long	city_pop	job	dob
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	0.000082	Jeff	115	M	351 Darlene Green	...	33.9659	-80.9355	0.114727	Mechanical engineer	1968-03-19
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	0.001267	Joanne	457	F	3638 Marsh Union	...	40.3207	-110.4360	0.000096	Sales professional, IT	1990-01-17
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	0.001769	Ashley	249	F	9333 Valentine Point	...	40.6729	-73.5365	0.011860	Librarian, public	1970-10-21

- Encoding the data into numerical values

```
[29] from sklearn.preprocessing import StandardScaler, LabelEncoder
     le = LabelEncoder()
     data['last'] = le.fit_transform(data['last'])
```

Train and Test Model

```
[34] # Split the data into training and testing sets
     from sklearn.model_selection import train_test_split
     X = data.drop('category', axis=1)
     y = data['category']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Decision Tree Algorithm

- Importing Important Libraries

```
[35] import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score
     from sklearn import tree
```

```
[36] print(data['is_fraud'].unique())

     [0 1]
```

- Choosing the required features

```
[37] X=data[['amt','city_pop']]
     y=data['is_fraud']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[38] clf = DecisionTreeClassifier(max_depth=3,criterion="entropy",random_state=100)
     clf.fit(X_train, y_train)

     DecisionTreeClassifier
     DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=100)
```

```
[39] y_pred=clf.predict(X_test)
```

```
[40] df=pd.DataFrame(y_test,y_pred)
```

```
[41] df=pd.DataFrame({"Actual application type":y_test,'Predicted application type':y_pred})
```

- The actual and the predicted values are shown here

```
df
```

	Actual application type	Predicted application type
119106	0	0
179292	0	0
540729	0	0
374360	0	0
314574	0	0
...	...	...
444284	0	0
89444	0	0
298536	0	0
301993	0	0
354010	1	0

111144 rows × 2 columns

- The accuracy of the model is as follows:

```
[43] accuracy_score(y_test,y_pred)*100
```

```
99.61671345281796
```

Naive Bayes Algorithm

- Choosing the required features:-

Naive Bayes

```
print(data['is_fraud'].unique())
```

```
[0 1]
```

```
[62] X=data[['amt','city_pop']]
y=data['is_fraud']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X=data[['amt','city_pop']]
y=data['is_fraud']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 0)
```

```
[65] from sklearn.naive_bayes import GaussianNB
clf1=GaussianNB()
clf1.fit(X_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

- Accuray of the model:-

```
from sklearn.metrics import classification_report, accuracy_score
y_pred1=clf1.predict(X_test)
score=accuracy_score(y_pred1,y_test)
print(score)
```

```
0.9922766861009141
```

- Actual and predicted values:-

```
[67] df1=pd.DataFrame(y_test,y_pred1)
     df1=pd.DataFrame({"Actual":y_test,'Predicted':y_pred1})
     print(df1)

             Actual  Predicted
     102051       0          0
     270705       0          0
     308877       0          0
     465523       0          0
     358267       0          0
     ...        ...        ...
     353837       0          0
     249257       0          0
     107904       0          0
     64537        0          0
     176736       0          0

     [138930 rows x 2 columns]
```

Model Performance Measures

- Now find the root mean squared error which takes the difference between your model's predictions and the ground truth, square it, and average it out across the whole dataset.

```
mse = mean_squared_error(y_test, y_pred)
print('Mean squared error:', mse)

Mean squared error: 0.003670718888785162
```

```
[100] r2 = r2_score(y_test, y_pred)
      print('R^2 score:', r2)

      R^2 score: 0.03861942949538144
```

In the following code we calculated root mean squared error and Mean absolute error where Root mean squared error calculates the transformation between values predicted by a model and actual values and Mean Absolute Error finds the magnitude of difference between the prediction of an observation and the true value of that observation.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pandas as pd
import numpy as np

# Create linear regression model
model = LinearRegression()

# Train model on training set
model.fit(X_train, y_train)

# Predict output for test set
y_pred = model.predict(X_test)

# Calculate performance measures
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)

# Print performance measures
print('Root mean squared error:', rmse)
print('Mean absolute error:', mae)
```

```
Root mean squared error: 0.06058645796533382
Mean absolute error: 0.007827903862067456
```

In the following code Precision gives the quality of a positive prediction made by the model and Recall showcases the percentage of data samples that a machine learning model correctly identifies as belonging to a class of interest.

```
from sklearn.metrics import precision_score, recall_score
import pandas as pd
import numpy as np

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print('Precision:', precision)
print('Recall:', recall)
```

```
Precision: 0.5363636363636364
Recall: 0.27699530516431925
```

```
[116] from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import confusion_matrix
      import pandas as pd
      import numpy as np

      # Create model
      model = RandomForestClassifier()

      # Train model on training set
      model.fit(X_train, y_train)

      # Predict output for test set
      y_pred = model.predict(X_test)

      # Calculate confusion matrix
      cm = confusion_matrix(y_test, y_pred)

      # Print confusion matrix
      print('Confusion matrix:\n', cm)
```

```
Confusion matrix:
 [[110616    102]
 [   308    118]]
```

Result and Analysis

Comparison of Algorithms:

Algorithm	**Accuracy**
Naive bayes	99.23%
Decision Tree	99.61%

- Decision Tree has performed better than Naive Bayes in our case.
- Decision trees are known for their interpretability and flexibility. They can handle a wide range of data types.
- Decision trees can overfit the training data if the tree is too complex, and may not generalize well to new data.

Conclusion

Fraud detection systems are critical tools in the financial industry for detecting and preventing fraudulent transactions. These systems use advanced technologies such as machine learning algorithms, big data analytics, and artificial intelligence to analyze large volumes of data in real-time.By analyzing historical transaction data, fraud detection systems can identify patterns and anomalies that may indicate fraudulent activity.

These systems can also incorporate various data sources, such as social media and external databases, to enhance the accuracy of the fraud detection process.The effectiveness of fraud detection systems depends on the quality of the data used, the accuracy of the algorithms, and the ability to detect new and evolving fraud patterns.

Therefore, continuous monitoring and improvement of these systems are essential.
Overall, fraud detection systems play a crucial role in mitigating financial losses due to fraudulent activities and maintaining the trust and confidence of customers in the financial industry. With the increasing sophistication of fraudsters, it is important to continue to develop and refine fraud detection systems to stay ahead of the constantly evolving threat landscape.