NAAN MUDHALAVAN : IBM

PHASE : 4

DEVELOPMENT PART 2

TECHNOLOGY : DATA SCIENCE

PROJECT TITLE : CREDIT CARD FRAUD DETECTION

Topic:

continue bulding the project by performing different activities like feature engineering, model training, evaluation etc as per the instructions in the project.

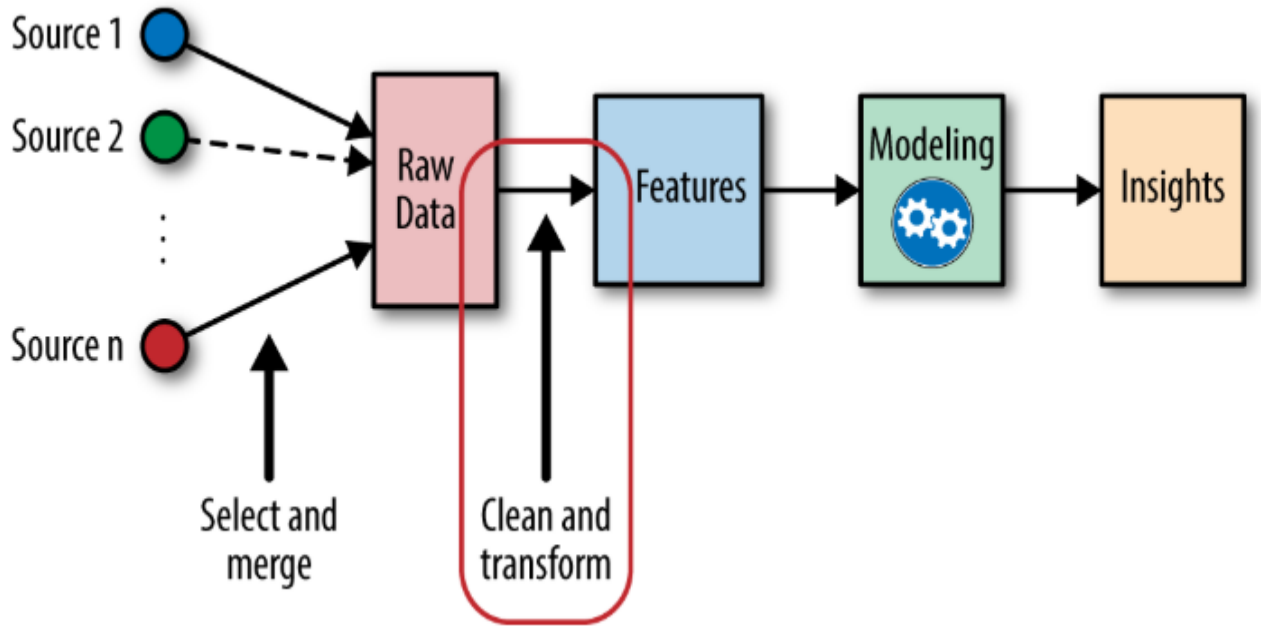
Introduction:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. As you may know, a “feature” is any measurable input that can be used in a predictive model — it could be the color of an object or the sound of someone’s voice. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches

Feature engineering:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it

might be necessary to design and train better features. As you may know, a “feature” is any measurable input that can be used in a predictive model — it could be the color of an object or the sound of someone’s voice. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches.



Data set:

```
import pandas
data = pandas.read_csv("creditcard.csv")
print(data)
```

Output:

	Time	V1	...	Amount	Class
0	0.0	-1.359807	...	149.62	0
1	0.0	1.191857	...	2.69	0

2	1.0	-1.358354	...	378.66	0
3	1.0	-0.966272	...	123.50	0
4	2.0	-1.158233	...	69.99	0
...	...	...	...	...	...
284802	172786.0	-11.881118	...	0.77	0
284803	172787.0	-0.732789	...	24.79	0
284804	172788.0	1.919565	...	67.88	0
284805	172788.0	-0.240440	...	10.00	0
284806	172792.0	-0.533413	...	217.00	0

[284807 rows x 31 columns]

Feature engineering various process :

- Feature creation
- Transformation
- Feature extraction
- Exploratory data analysis
- Benchmark

Feature Creation:

Creating features involves creating new variables which will be most helpful for our model..

Transformations:

Feature transformation is simply a function that transforms features from one representation to another.

Program:

```
import matplotlib.pyplot as plt


labels = ["Genuine", "Fraud"]

count_classes = dataframe.value_counts(dataframe['Class'], sort= True)

count_classes.plot(kind = "bar", rot = 0)

plt.title("Visualization of Labels")

plt.ylabel("Count")

plt.xticks(range(2), labels)

plt.show()
```

OUTPUT:

```
print("Number of Genuine transactions: ", non_fraud)
print("Number of Fraud transactions: ", fraud)
print("Percentage of Fraud transactions: {:.4f}".format(fraud_percent))
```

```
Number of Genuine transactions:  284315
Number of Fraud transactions:  492
Percentage of Fraud transactions: 0.1727
```

In [7]:
```
# Visualize the "Labels" column in our dataset

labels = ["Genuine", "Fraud"]
count_classes = dataframe.value_counts(dataframe['Class'], sort= True)
count_classes.plot(kind = "bar", rot = 0)
plt.title("Visualization of Labels")
plt.ylabel("Count")
plt.xticks(range(2), labels)
plt.show()
```

Feature Extraction:

Feature extraction is the process of extracting features from a data set to identify useful information.

PROGRAM:

```
# X_tsfresh contains the extracted tsfresh features
X_tsfresh = extract_features(...)

# which are now filtered to only contain relevant features
X_tsfresh_filtered = some_feature_selection(X_tsfresh, y, ....)

# we can easily construct the corresponding settings object
kind_to_fc_parameters
=tsfresh.feature_extraction.settings.from_columns(X_tsfresh_filtered)
```

OUTPUT:

```
Wall time: 393 ms

('Accuracy score: 0.9590643274853801',
'Recall score: 0.9722222222222222',
'Precision score: 0.963302752293578')
    },
```

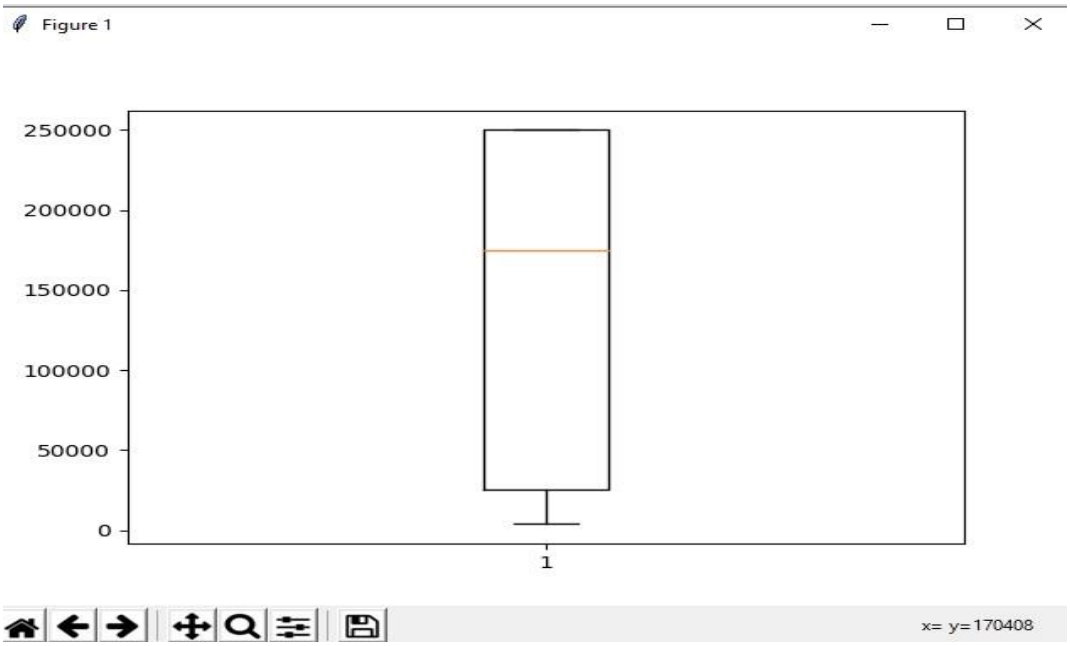
Exploratory Data Analysis :

Exploratory data analysis (EDA) is a powerful and simple tool that can be used to improve your understanding of your data, by exploring its properties.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
DF = pd.read_csv("https://raw.githubusercontent.com / fivethirtyeight / data / master / airline-safety / airline-safety.csv")
y = list(DF.population)
plt.boxplot(y)
plt.show()
```

OUTPUT:



Benchmark :

A Benchmark Model is the most user-friendly, dependable, transparent, and interpretable model against which you can measure your own.

Feature engineering techniques:

Lets see a few feature engineering best techniques that you can use. Some of the techniques listed may work better with certain algorithms or datasets, while others may be useful in all situations.

- Imputation
- Handling outliers
- Log transform
- One hot encoding
- Scaling

Imputation:

When it comes to preparing your data for machine learning, missing values are one of the most typical issues.

There are two types of Imputation

1.Numerical Imputation

```
#Filling all missing values with 0
data = data.fillna(0)
```

2. categorical Imputation

```
#Max fill function for categorical columns
```

PROGRAM:

```
     miss_mean_imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
 miss_mean_imputer = miss_mean_imputer.fit(df)
imputed_df = miss_mean_imputer.transform(df.values) print(imputed_df)
```

OUTPUT:

	C0	C1
0	0.2601	0.7154
1	0.2358	NaN
2	0.1429	0.2615
3	0.1259	0.5846
4	0.7526	NaN
5	0.7341	0.8308
6	0.4546	0.4962
7	0.1426	NaN
8	0.1490	0.5340
9	0.2500	0.6731

Handling outliers:

Outlier handling is a technique for removing outliers from a dataset. This method can be used on a variety of scales to produce a more accurate data representation. This has an impact on the model's performance**.**

PROGRAM:

```
# Box Plot
import seaborn as sns
sns.boxplot(df_diabetics['bmi'])
```

OUTPUT:

Outliers present in the bmi columns

Log Transform

Log Transform is the most used technique among data scientists. It's mostly used to turn a skewed distribution into a normal or less-skewed distribution. We take the log of the values in a column and utilise those values as the column in this transform.

```
//Log Example
df[log_price] = np.log(df['Price'])
```

PROGRAM:

```
import numpy as np

# Input data
data = [1, 2, 3, 4, 5]

# Log transformation
transformed_data = np.log(data)
```

```
# Output
print("Original Data:", data)
print("Log-Transformed Data:", transformed_data)
```

OUTPUT:

Original Data: [1, 2, 3, 4, 5]

Log-Transformed Data: [0.69314718 1.09861229 1.38629436 1.60943791]

One-hot encoding:

A one-hot encoding is a type of encoding in which an element of a finite set is represented by the index in that set, where only one element has its index set to “1” and all other elements are assigned indices within the range [0, n-1].

PROGRAM:

```
import pandas as pd

# List of categorical values
categories = ['Red', 'Green', 'Blue', 'Red', 'Green']

# Create a DataFrame and perform one-hot encoding
df = pd.get_dummies(categories)
```

```
# Output
print("Original Data:")
print(categories)
print("\nOne-Hot Encoded Data:")
print(df)
```

OUTPUT:

```
Original Data:
['Red', 'Green', 'Blue', 'Red', 'Green']

One-Hot Encoded Data:
    Blue  Green    Red
0  False  False   True
1  False   True  False
2   True  False  False
3  False  False   True
4  False   True  False
```

SCALING:

Feature scaling is one of the most pervasive and difficult problems in machine learning, yet it's one of the most important things to get right.In order to train a predictive model, we need data with a known set of features that needs to be scaled.

There are two types in scaling

- Normalization
- Standardization

NORMALIZATION:

```
from sklearn import preprocessing
import numpy as np
x_array = np.array([2,3,5,6,7,4,8,7,6])
normalized_arr = preprocessing.normalize([x_array])
print(normalized_arr)
```

OUTPUT:

```
[0.11785113, 0.1767767 , 0.29462783, 0.35355339, 0.41247896,
       0.23570226, 0.47140452, 0.41247896, 0.35355339]
```

STANDARDIZATION TECHNIQUE:

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

dataset = load_iris()
object= StandardScaler()
```

```
# Splitting the independent and dependent variables
```

OUTPUT:

```
[[-9.00681170e-01  1.01900435e+00 -1.34022653e+00 -1.31544430e+00]
 [-1.14301691e+00 -1.31979479e-01 -1.34022653e+00 -1.31544430e+00]
 [-1.38535265e+00  3.28414053e-01 -1.39706395e+00 -1.31544430e+00]
 [-1.50652052e+00  9.82172869e-02 -1.28338910e+00 -1.31544430e+00]
 [-1.02184904e+00  1.24920112e+00 -1.34022653e+00 -1.31544430e+00]
 [-5.37177559e-01  1.93979142e+00 -1.16971425e+00 -1.05217993e+00]
 [-1.50652052e+00  7.88807586e-01 -1.34022653e+00 -1.18381211e+00]
 [-1.02184904e+00  7.88807586e-01 -1.28338910e+00 -1.31544430e+00]
 [-1.74885626e+00 -3.62176246e-01 -1.34022653e+00 -1.31544430e+00]
 [-1.14301691e+00  9.82172869e-02 -1.28338910e+00 -1.44707648e+00]
 [-5.37177559e-01  1.47939788e+00 -1.28338910e+00 -1.31544430e+00]
 [-1.26418478e+00  7.88807586e-01 -1.22655167e+00 -1.31544430e+00]
 [-1.26418478e+00 -1.31979479e-01 -1.34022653e+00 -1.44707648e+00]
 [-1.87002413e+00 -1.31979479e-01 -1.51073881e+00 -1.44707648e+00]
 [-5.25060772e-02  2.16998818e+00 -1.45390138e+00 -1.31544430e+00]
 [-1.73673948e-01  3.09077525e+00 -1.28338910e+00 -1.05217993e+00]
 [-5.37177559e-01  1.93979142e+00 -1.39706395e+00 -1.05217993e+00]
 [-9.00681170e-01  1.01900435e+00 -1.34022653e+00 -1.18381211e+00]
 [-1.73673948e-01  1.70959465e+00 -1.16971425e+00 -1.18381211e+00]
 [-9.00681170e-01  1.70959465e+00 -1.28338910e+00 -1.18381211e+00]
```

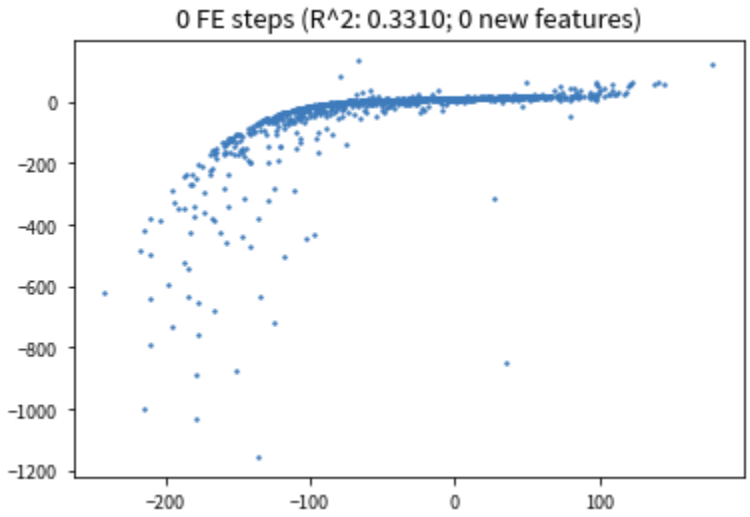
Feature Engineering Tools:

There are many tools which will help you in automating the entire feature engineering process and producing a large pool of features in a short period of time for both classification and regression tasks.

AutoFeat:

AutoFeat helps to perform Linear Prediction Models with Automated Feature Engineering and Selection. AutoFeat allows you to select the units of the input variables in order to avoid the construction of physically nonsensical features.

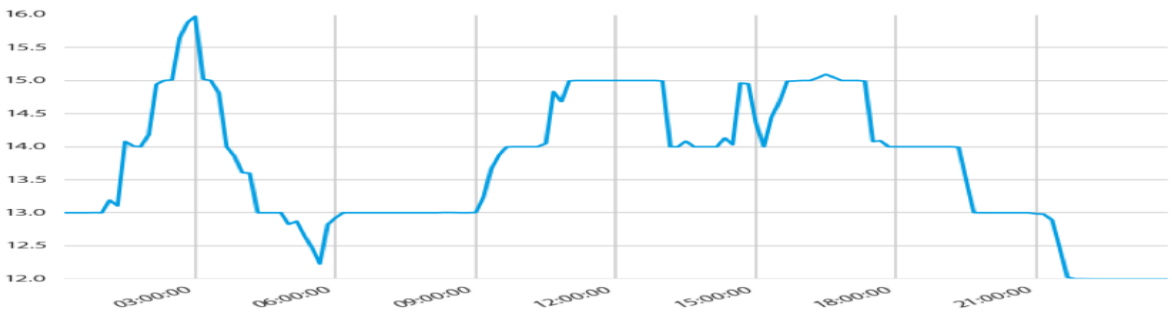
EXAMPLE:



TsFresh:

TsFresh is a python package. It calculates a huge number of time series characteristics, or features, automatically. In addition, the package includes methods for assessing the explanatory power and significance of such traits in regression and classification tasks.

EXAMPLE:



ONEBM :

OneBM interacts directly with a database's raw tables. It slowly joins the tables, taking different paths on the relational tree. It recognises simple data types in the joint results and applies pre-defined feature engineering.

EXPLORE KIT:

Based on the idea that extremely informative features are typically the consequence of manipulating basic ones, ExploreKit identifies common operators to alter each feature independently or combine multiple of them. Instead of running feature selection on all developed features.

CONCLUSION:

We investigated the data, checking for data unbalancing, visualizing the features, and understanding the relationship between different features. We then investigated two predictive models. The data was split into three parts, a train set, a validation set, and a test set. For the first three models, we only used the train and test set.