Machine Learning
Digital Assignment-1

Name: Ashish David John
Reg: 22BIT0188

Q. Study the Various Feature Selection and Data Pre-processing Techniques used in Machine Learning and Implement in Python.

Data Preprocessing

Data preprocessing involves performing several important tasks to the data set before applying machine learning or deep learning techniques or developing models on it.
The tasks include:
- Exploring the data and its features
- Understanding the relationships between the features using visualizations and plots
- Handling issues within the data like missing values, outliers, un-encoded data,redundant fields etc.
- Feature selection, extraction and construction if necessary

Data preprocessing involves handling the above mentioned tasks through a variety of techniques. There are four major tasks involved in preprocessing, they are:

1. **Data Cleaning**
   Real world data can be incomplete, noisy or inconsistent due to many factors that affect data collection such as inaccurate instruments, incorrect form input, etc. Data Cleaning methods attempt to fill

in missing values, smooth out noise while identifying outliers and correct inconsistencies in the data.

Missing values for example can be handled in several ways:

- **Ignoring the tuple**, i.e that particular data record with the missing values can be excluded from the dataset.

- **Filling in the missing values** with measures of central tendency like mean or median.

- **Use the most probable value** to fill the missing value with techniques like regression or decision tree induction or other induction methods

Noisy data can be smoothed out by **binning methods** or even by **regression**. Binning groups values into bins and replaces the individual value with representative values such as mean, median or boundary values. It helps reduce the randomness or noise in the data.

2. **Data Integration**

   Data integration is the merging of data from multiple data stores. However careless integration can lead to redundancy and inconsistency in the resulting data

set. Careful integration will help reduce these problems and help the model identify meaningful relationships from the data.
Technique involved in careful integration is:

- **Correlation Analysis**
  Chi-square Test for nominal data and Correlation coefficient and covariance for numeric attributes.

3. **Data Reduction**
   Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. Data reduction strategies include:
   - **Dimensionality reduction**: The process of reducing the number of random variables or attributes under consideration, while retaining as much information as possible. **PCA (Principal Component Analysis)** and **LDA (Linear Discriminant Analysis)** are some methods which transform the original data into a smaller (lower dimension) space.

   - **Numerosity reduction**: Replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or nonparametric. Ex: Regression model, log-linear model, histograms, clusterin, data cube aggregation.

4. **Data Transformation and Discretization**
   In data transformation, the data are transformed or consolidated into forms appropriate for tasks like model development or data mining.Techniques include:
   - **Normalization**: Use Min-Max Scaling or Z-score standardization to scale the data

   - **Encoding Categorical data**: Use One-Hot Encoding or Label Encoding, which converts categorical variables into binary vectors or assigns a unique integer to each category respectively.

   Discretization is where the raw values of a numeric attribute (age) are replaced by interval labels (0–10, 11–20, etc.) or conceptual labels (youth, adult, senior). Discretization can be performed using:
   - Binning
   - HIstogram Analysis
   - Cluster Analysis
   - Decision tree Analysis
   - Correlation Analysis

Feature Selection

Feature selection involves selecting a subset of the original features that are most relevant to the problem at hand. The goal is to reduce the dimensionality of the data set while retaining the most important features relevant to the class variable (dependent variable).

There are 3 approaches to feature selection:

1. **Filter Approach**

   This approach employs statistical measures to select features. Common statistical test conducted as part of this approach are:
   - Pearson's Correlation Analysis
   - Chi-Square Test
   - Information gain
   - Variance Threshold

2. **Wrapper Approach**

   In this approach, the selection of features is considered a search problem, in which different combinations of features are made, evaluated and compared. It provides an optimal set of features for training the model. Techniques include:
   - Forward Selection
   - Backward Selection
   - Exhaustive Feature Selection
   - Recursive Feature Elimination

3. **Embedded Approach**
   In this approach, the feature selection algorithm is blended as part of the learning algorithm. During the training step, the classifier adjusts its internal parameters, where the feature selection is integrated, and determines the appropriate weights/importance given for each feature. Algorithms and techniques include:
   - Regularization
   - Artificial Neural Network Weights
   - Decision Tree Importance Scores

Python Implementation

Dataset: loan_prediction.csv

Importing Python libraries:

Pandas: Used for data manipulation and analysis (e.g., reading, cleaning, and transforming data).
NumPy: Provides support for numerical operations and efficient handling of arrays/matrices.
Seaborn: Built on Matplotlib, used for advanced statistical visualizations (e.g., heatmaps, box plots).
Matplotlib: Core library for creating static, animated, and interactive plots in Python.

```
[78]: # import the Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading data set:

614 Records

13 Features

```
[80]: # Read the dataset
      df=pd.read_csv('loan_prediction.csv')

[82]: df.head()
```

[82]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicai
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

```
[84]: df.shape

[84]: (614, 13)
```

Statistics on the features

```
[86]: #Descriptive statistics
      df.describe()
```

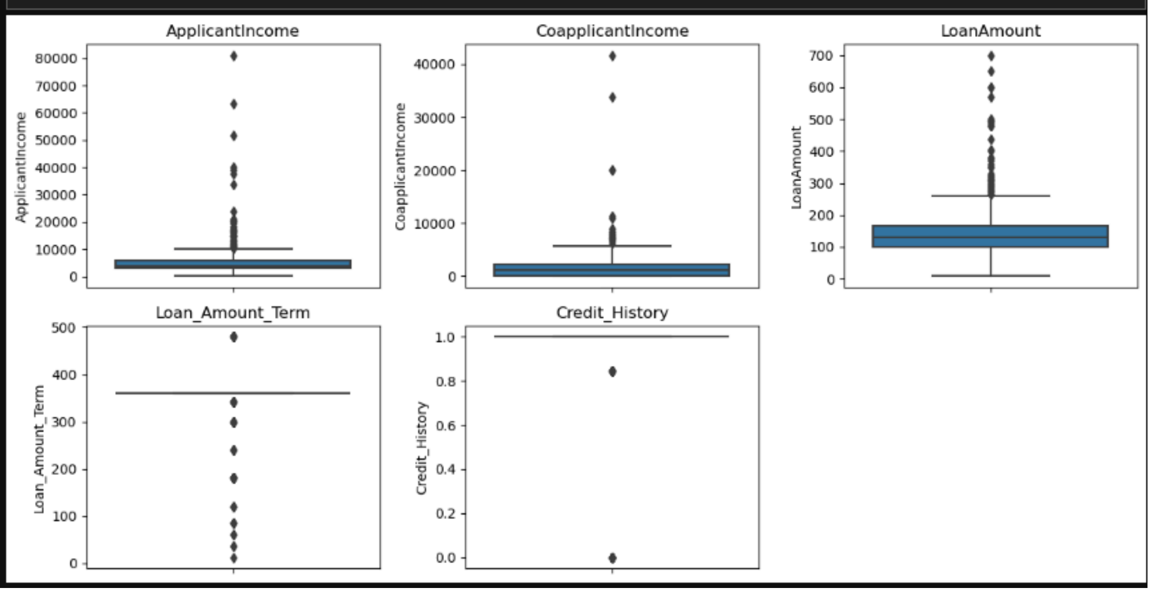
[86]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

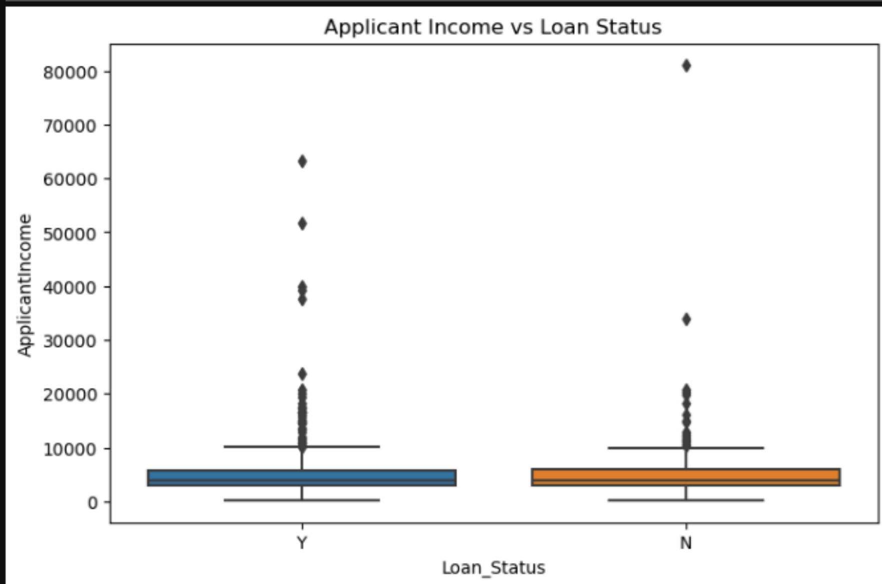
```
[16]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Box plots:

```
[216]: numeric_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History']

plt.figure(figsize=(12, 6))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(col)

plt.tight_layout()
plt.show()
```



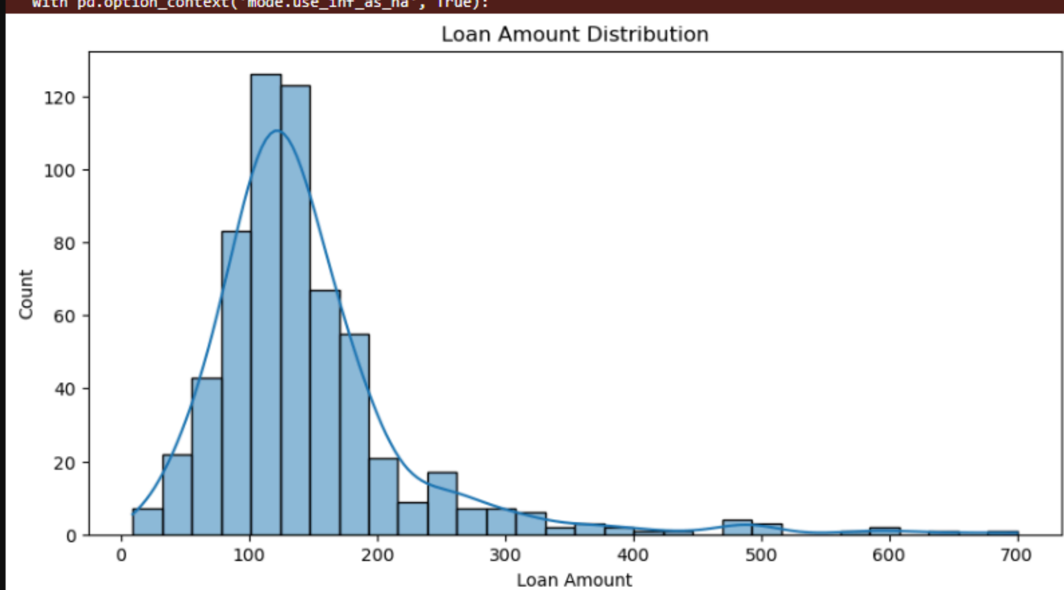
```
[217]: plt.figure(figsize=(8, 5))
       sns.boxplot(x='Loan_Status', y='ApplicantIncome', data=df)
       plt.title('Applicant Income vs Loan Status')
       plt.show()
```



Histogram:

```
[220]: plt.figure(figsize=(10, 5))
       sns.histplot(df['LoanAmount'].dropna(), bins=30, kde=True)
       plt.title('Loan Amount Distribution')
       plt.xlabel('Loan Amount')
       plt.ylabel('Count')
       plt.show()
```

```
C:\Users\altas\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and wil
l be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Dropping un-correlated columns

```
[20]: #drop unwanted columns
      df.drop(columns=['Loan_ID','Gender','Dependents','Self_Employed'], inplace=True)

[22]: df.head()
```

[22]:

	Married	Education	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	I
0	No	Graduate	5849	0.0	NaN	360.0	1.0	
1	Yes	Graduate	4583	1508.0	128.0	360.0	1.0	
2	Yes	Graduate	3000	0.0	66.0	360.0	1.0	
3	Yes	Not Graduate	2583	2358.0	120.0	360.0	1.0	
4	No	Graduate	6000	0.0	141.0	360.0	1.0	

```
[24]: df.isnull().sum()

[24]: Married               3
      Education             0
      ApplicantIncome       0
      CoapplicantIncome     0
      LoanAmount           22
      Loan_Amount_Term     14
      Credit_History       50
      Property_Area         0
      Loan_Status           0
      dtype: int64
```

Handling missing values

```
[26]: # Hndling Null Values
      df['Married'].fillna('Yes', inplace=True)
      df["LoanAmount"].fillna(df['LoanAmount'].mean(),inplace=True)
      df["Loan_Amount_Term"].fillna(df['Loan_Amount_Term'].mean(),inplace=True)
      df["Credit_History"].fillna(df['Credit_History'].mean(),inplace=True)

[28]: df.isnull().sum()

[28]: Married              0
      Education            0
      ApplicantIncome      0
      CoapplicantIncome    0
      LoanAmount           0
      Loan_Amount_Term     0
      Credit_History       0
      Property_Area        0
      Loan_Status          0
      dtype: int64
```

Determining independent and dependent variables

```
[32]: x=df.drop('Loan_Status',axis=1)
      y=df['Loan_Status']
```

Employing OneHot Encoder on X for columns 0,1,7

```
[36]: from sklearn.compose import ColumnTransformer
      from sklearn.preprocessing import OneHotEncoder

[38]: ct=ColumnTransformer([('oh',OneHotEncoder(),[0,1,7])],remainder='passthrough')

[40]: x=ct.fit_transform(x)
```

```
[44]: x

[44]: array([[  1.        ,   0.        ,   1.        , ..., 146.41216216,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , ..., 128.        ,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , ...,  66.        ,
         360.        ,   1.        ],
       ...,
       [  0.        ,   1.        ,   1.        , ..., 253.        ,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , ..., 187.        ,
         360.        ,   1.        ],
       [  1.        ,   0.        ,   1.        , ..., 133.        ,
         360.        ,   0.        ]])
```

Employing label encoder on Y

```
[46]: from sklearn.preprocessing import LabelEncoder

[48]: le=LabelEncoder()

[50]: y=le.fit_transform(y)

[52]: y

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

Scaling the data in x with StandardScaler

```
[54]: from sklearn.preprocessing import StandardScaler

[56]: sc=StandardScaler()

[58]: x=sc.fit_transform(x)

[60]: x

[60]: array([[ 1.37208932e+00, -1.37208932e+00,  5.28362249e-01, ...,
          3.38478577e-16,  2.79850543e-01,  4.51640451e-01],
        [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, ...,
         -2.19273315e-01,  2.79850543e-01,  4.51640451e-01],
        [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, ...,
         -9.57640999e-01,  2.79850543e-01,  4.51640451e-01],
        ...,
        [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, ...,
          1.26937121e+00,  2.79850543e-01,  4.51640451e-01],
        [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, ...,
          4.83366900e-01,  2.79850543e-01,  4.51640451e-01],
        [ 1.37208932e+00, -1.37208932e+00,  5.28362249e-01, ...,
         -1.59727534e-01,  2.79850543e-01, -2.41044061e+00]])
```

Splitting into training and test data for Random Forest Classifier to look for important features

```
[154]: from sklearn.model_selection import train_test_split
       x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)

[156]: #Create Model
       from sklearn.tree import DecisionTreeClassifier

[158]: model=DecisionTreeClassifier(criterion='entropy',random_state=0)

[160]: model.fit(x_train,y_train)

[160]:          DecisionTreeClassifier
       DecisionTreeClassifier(criterion='entropy', random_state=0)

[190]: x_train_df = pd.DataFrame(x_train)
       feature_importance = model.feature_importances_
       importance_df = pd.DataFrame({ 'Importance': model.feature_importances_})
       importance_df = importance_df.sort_values(by='Importance', ascending=False)

       print(importance_df)

           Importance
       7     0.251841
       11    0.239336
       8     0.193390
       9     0.184655
       10    0.038612
       5     0.027339
       3     0.015267
       4     0.013869
       6     0.013746
       0     0.011302
       2     0.008170
       1     0.002474
```

Conducting PCA on features to reduce dimensions

```
[200]: from sklearn.decomposition import PCA
       # Apply PCA
       pca = PCA(n_components=5)  # Select number of principal components
       x_train_pca = pca.fit_transform(x_train_df)
       x_train_pca_df = pd.DataFrame(x_train_pca, columns=[f'PC{i+1}' for i in range(x_train_pca.shape[1])])

       print("variance ratio:", pca.explained_variance_ratio_)

       print(x_train_pca_df.head())
```

```
variance ratio: [0.1917815  0.16652185 0.13429331 0.12700357 0.111751  ]
        PC1       PC2       PC3       PC4       PC5
0  0.833641 -0.216835  1.939970 -0.401967  0.083385
1  0.259060  2.152654  0.156454 -1.515134 -1.217731
2  0.640298 -0.778399 -0.759435 -0.422961  1.885358
3 -0.570372  1.876806 -0.634194  1.898792 -0.705169
4  0.974812 -0.194645  1.732810 -0.564390 -0.139657
```

References:

1. Data Mining – Concepts and Techniques – 3rd Edition, Jiawei Han, Micheline Kamber & Jian Pei-Elsevier

2. https://www.stratascratch.com/blog/feature-selection-techniques-in-machine-learning/