BIG DATA COURSE WORK REPORT- GROUP_010

Module

CO7093 Big Data and Predictive Analytics

Assignment-

Coursework

Part 1. Building up a basic predictive model

1.1 Data cleaning and transformation

1.1.1 Loading dataset to pandas & it's shape

Loaded the dataset (patients.csv) to pandas data frame. Checked the data frame for inconsistencies and found missing values. The shape of the data was checked to ensure the total number of columns and rows.

Fig. 1.1-1 Shape of columns before and after data cleaning and transformation

Shape before cleaning

```
(200031, 22)
```

1.1.2 Removing irrelevant columns & rows with null values

Columns such as 'index', 'USMER', 'MEDICAL_UNIT', 'PATIENT_TYPE', 'DATE_DIED', 'PREGNANT' were removed during preprocessing. These columns were removed because they were identified as not directly significant in predicting the ICU admission and/or majority of null values across column. Initially, We also tried to filter the datset based on DATE_DIED column to exclude patients that are already dead from the dataframe. However, this approach significantly reduced the size of the patient's dataset. As a result, we opted to drop the entire column instead, preserving the overall sample size for modelling.

1.1.3 Handling missing values

Non-numeric placeholders such as (’?’) was replaced with NaN. Summary statistics were generated before and after to validate the results. All the rows that had missing values were removed from the dataset. The shape of the dataset was checked again.

Fig. 1.1-2 Shape after cleaning

```
(189586, 16)
```

1.1.4 Data type adjustment

Fig. 1.1-3 Data type before adjustment

```
index                    int64
USMER                    int64
MEDICAL_UNIT             int64
SEX                      int64
PATIENT_TYPE             int64
DATE_DIED               object
INTUBED                 object
PNEUMONIA               object
AGE                      int64
PREGNANT                object
DIABETES                object
COPD                    object
ASTHMA                  object
INMSUPR                 object
HIPERTENSION            object
OTHER_DISEASE           object
CARDIOVASCULAR          object
OBESITY                 object
RENAL_CHRONIC           object
TOBACCO                 object
CLASIFFICATION_FINAL     int64
ICU                     object
dtype: object
```

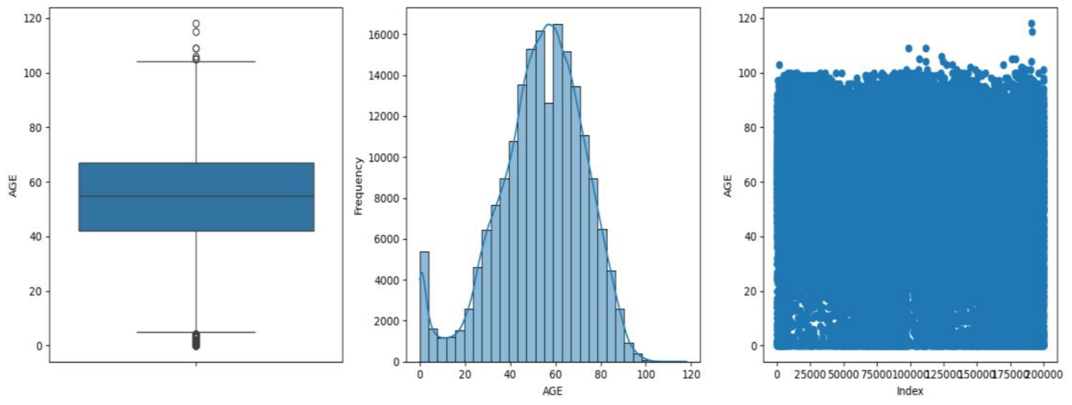
As shown in Fig. 1.1-3 several columns have been inappropriately stored as an ‘object’ even though they contain numeric data. To rectify this, the object datatype was changed to integer (Int) datatype that can store null values. We also considered changing the datatype to ‘categorical features’ but found this approach less effective, so we opted to convert all columns to integers to maintain consistency.

1.1.5 Handling outliers using IQR method

Outlier detection was focused on 'Age' column as that was the only feature with continuous values. Binary columns (primarily has 1s and 2s) were excluded in this as they do not exhibit outlier behaviour by nature. Hence, 'Age' column was the only feature analysed for outlier.

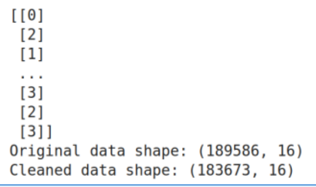
Visual tools like box plot, histogram and scatter plot were used to visualise the outliers as shown in fig. 1.1-4

Fig. 1.1-4 Visual analysis of outliers



IQR method was then used to identify and remove outliers. The Q1 and Q3 were calculated and any values falling below Q1 - 1.5*IQR or above Q3 + 1.5*IQR were classified as outliers. The dataset is checked again after outlier removal as shown in fig. 1.1-5.

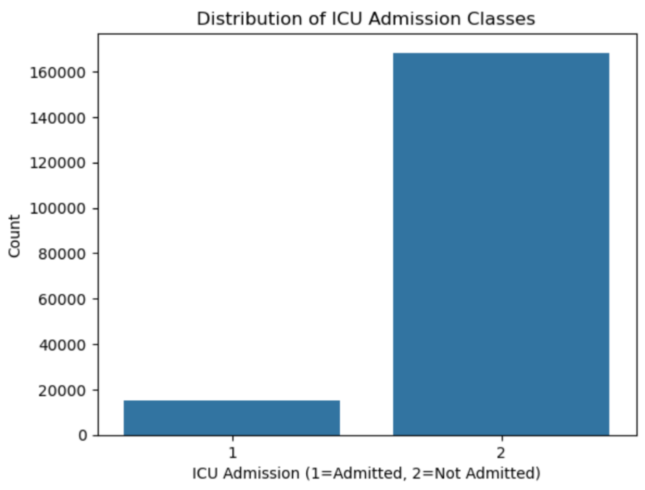
Fig. 1.1-4 shape after removing outliers



```
[[0]
 [2]
 [1]
 ...
 [3]
 [2]
 [3]]
Original data shape: (189586, 16)
Cleaned data shape: (183673, 16)
```

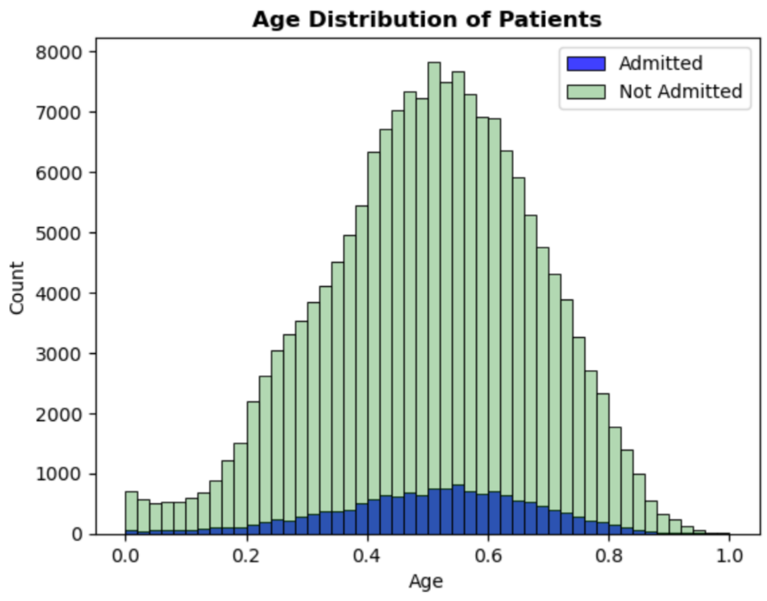
1.2 Data Visualisation

Fig. 1.2-1 Plot for distribution of unique classes of the target variable 'ICU'



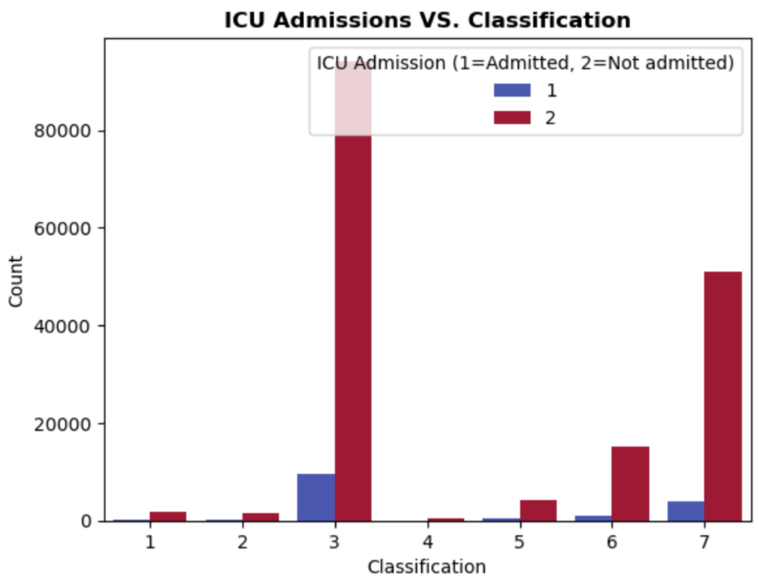
The above chart (Figure 1.2-1) portrays that the data is imbalanced and includes many not admitted patients.

Fig. 1.2.2 The count of ICU against the Age of Patients



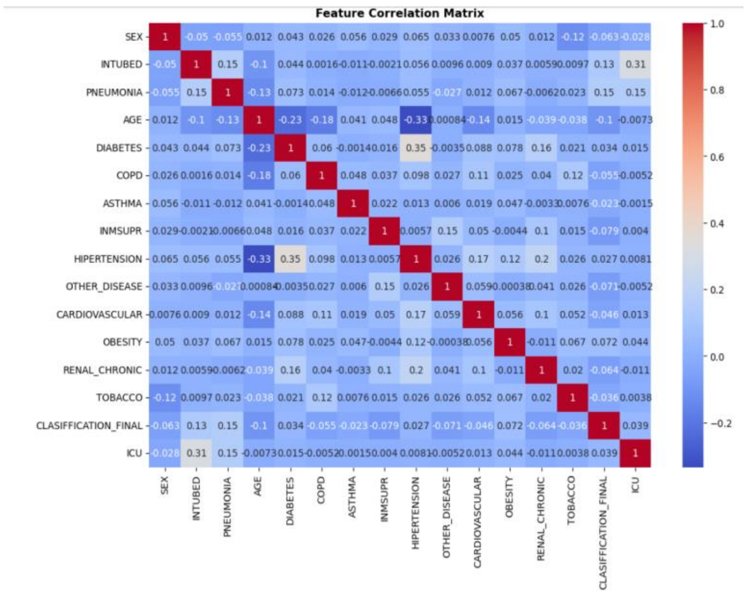
Based on Fig. 1.2-2 The age distribution of patients follows a normal (bell-shaped) pattern, with the majority falling within the mid-range of the normalized age values. ICU admissions (blue bars) are present across all age groups but appear more concentrated in the central age ranges. However, non-ICU cases (light green) significantly outnumber ICU admissions, indicating a strong class imbalance in the dataset.

Fig. 1.2-2 Plot a graph that displays the count of target variable against 'CLASIFFICATION_FINAL'.



As shown in Fig. 1.2-2 maximum patients were tested covid positive

Fig. 1.2--3 Correlation matrix



Based on the above correlation matrix (Fig. 1.2-3), it shows that there were not many highly correlated features. For this reason, we measured the threshold of ~+/-0.5 as highly correlated

Fig.1.2-4 Scatter matrix

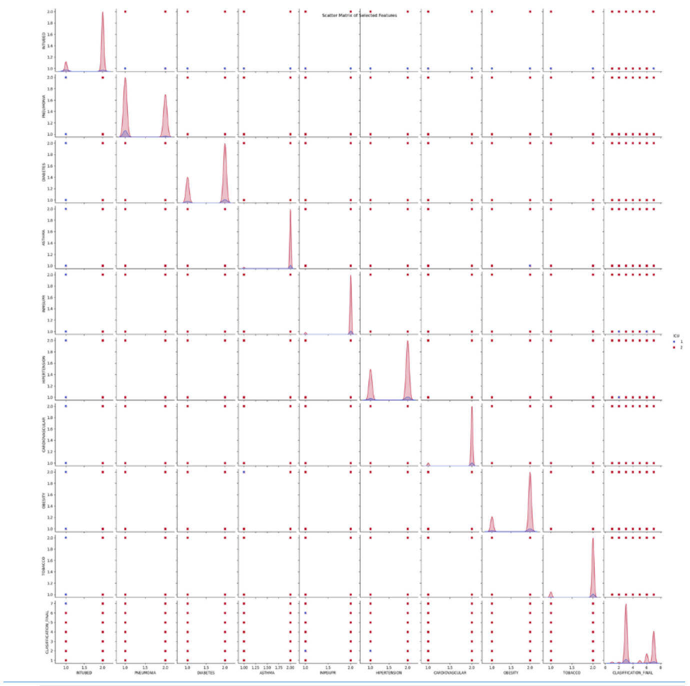


Fig.1.2-5 Correlation of ICU matrix against All feature importance

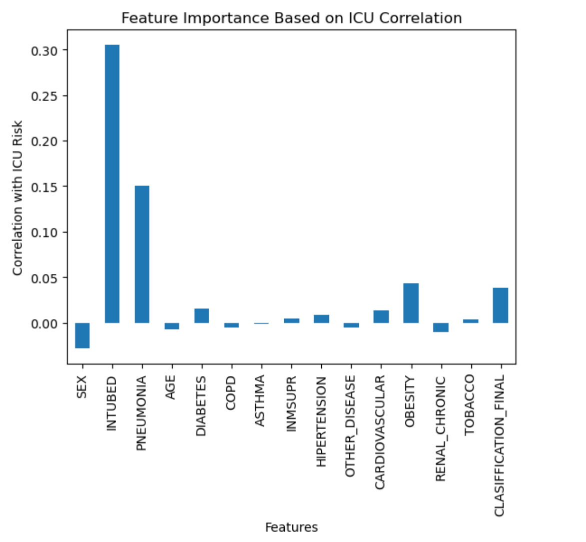


Fig.1.2-6 ICU administration risk against Diabetes and Hypertension

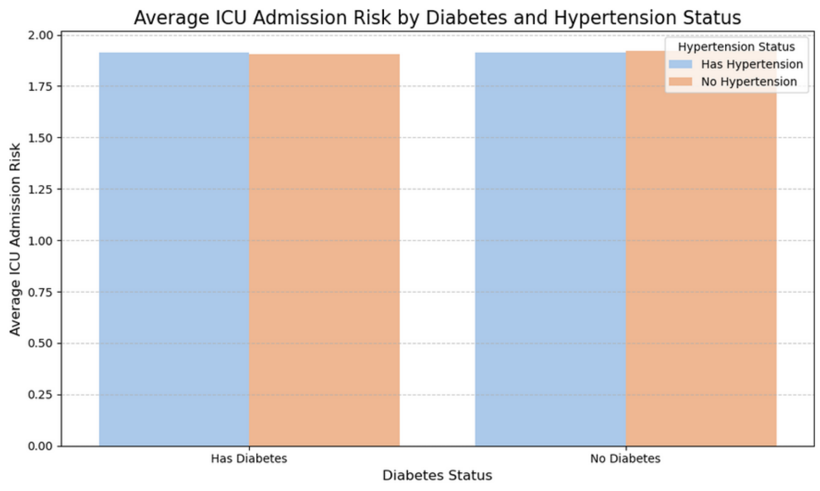
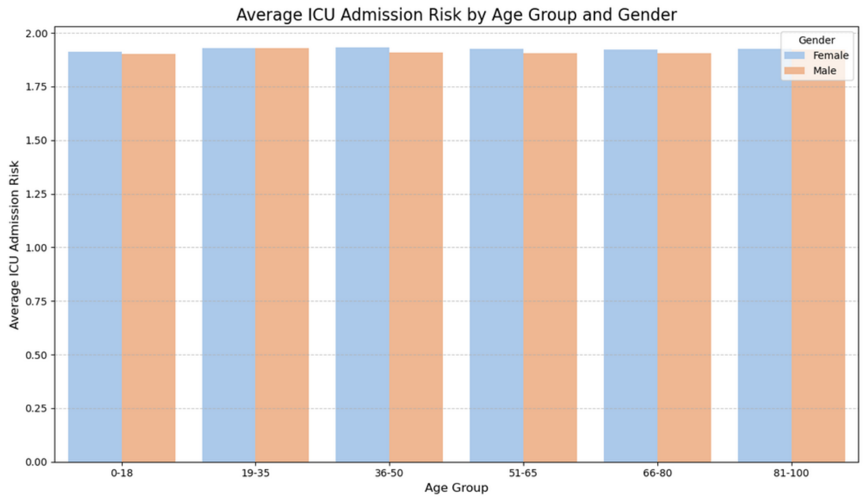


Fig.1.2-7 ICU administration risk against Age group and gender.



1.3 Model Building

We first selected the features we wanted to target for ICU prediction. The figure below (Fig. 1.3-1) shows the figures selected for model building for prediction.

Fig. 1.3-1 Features selected for model building

```
Selected features via RFE: Index(['INTUBED', 'PNEUMONIA', 'CARDIOVASCULAR', 'OBESITY', 'RENAL_CHRONIC'], dtype='object')
```

We then used Logistic regression and below is its Evaluation:

Fig. 1.3-2 Lineal model Accuracy and coefficients

```
Accuracy 0.9164275642037751
Intercept: [-2.5046601]
Coefficients:
             INTUBED: 1.9025319982494036
           PNEUMONIA: 1.1106528821180006
      CARDIOVASCULAR: 0.1559655460014761
             OBESITY: 0.22810201208443032
       RENAL_CHRONIC: -0.20792519887612065
```

Fig. 1.3-3 Training score

```
Training accuracy: 0.9165090037975201
Test accuracy: 0.9161018102626922
```

Fig. 1.3-4 Confusion matrix

```
Confusion Matrix:
[[    0  3082]
 [    0 33653]]
Classification Report:
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00      3082
         2.0       0.92      1.00      0.96     33653

    accuracy                           0.92     36735
   macro avg       0.46      0.50      0.48     36735
weighted avg       0.84      0.92      0.88     36735
```

Fig. 1.3-5 Cross validation

```
Cross-validation scores for each fold: [0.91643075 0.91643075 0.91643075 0.9164262  0.9164262  0.9164262
 0.9164262  0.9164262  0.9164262  0.9164262 ]
Mean Cross-validation Score: 0.9164275641517537
Standard Deviation of Scores: 2.0850569929329496e-06
```

Fig. 1.3-6 Confusion threshold before balancing the data

```
Confusion Matrix for threshold 0.05:
Predicted     1
Actual
1           214
2          6867

Confusion Matrix for threshold 0.1:
Predicted     1
Actual
1           214
2          6867

Confusion Matrix for threshold 0.2:
Predicted     1
Actual
1           214
2          6867
```

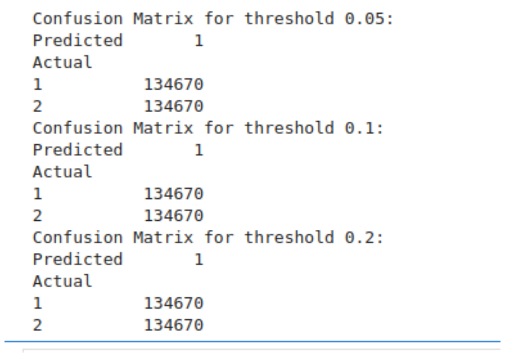
Fig. 1.3-7 Balancing data using SMOTE

```
Original dataset shape: (146938, 5), (146938,)
SMOTE balanced dataset shape: (269340, 5), (269340,)
```

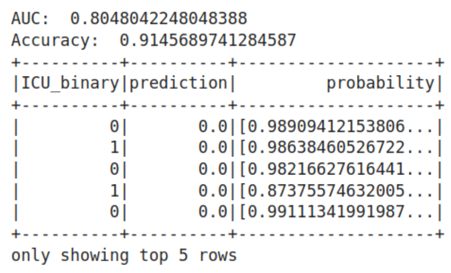
Fig. 1.3-8 Cross validation for balanced data

```
Cross-validation scores for each fold: [0.712668   0.70505681 0.70773001 0.70984629 0.70810128 0.71055172
 0.71422737 0.71162843 0.71032895 0.71229673]
Mean Cross-validation Score: 0.7102435583277641
Standard Deviation of Scores: 0.0025681083264160226
```

Fig. 1.3-9 Confusion threshold after balancing the data



```
Confusion Matrix for threshold 0.05:
Predicted       1
Actual
1          134670
2          134670
Confusion Matrix for threshold 0.1:
Predicted       1
Actual
1          134670
2          134670
Confusion Matrix for threshold 0.2:
Predicted       1
Actual
1          134670
2          134670
```



```
AUC:  0.8048042248048388
Accuracy:  0.9145689741284587
+----------+----------+--------------------+
|ICU_binary|prediction|         probability|
+----------+----------+--------------------+
|         0|       0.0|[0.98909412153806...|
|         1|       0.0|[0.98638460526722...|
|         0|       0.0|[0.98216627616441...|
|         1|       0.0|[0.87375574632005...|
|         0|       0.0|[0.99111341991987...|
+----------+----------+--------------------+
only showing top 5 rows
```

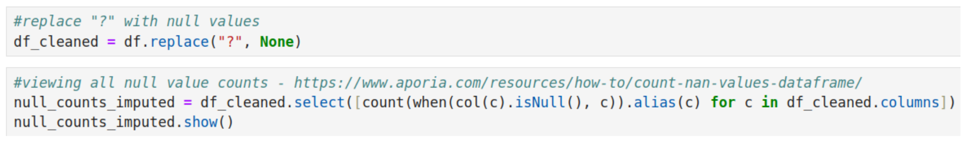
Evaluation of Results

Oversampling was used to try and balance out the data by using SMOTE, however the results varied. This is explained and reasoned below.

- The cross-validation scores dropped from 0.916 to 0.711 once balanced, this may suggest that there could be data leakage, or the model is overfitting to the training data.
- When using oversampling it can sometimes artificially balance the data set, this can result in an overrepresentation of certain feature from the minority class. This can result in high False positives or high false negatives.
- The accuracy of the 0.9145 shows that it predicts
- However, the AUC score indicated a good performance overall as it predicts over 91% of cases correctly. This is crucial for predictions.
- Additional metrics such as the F1-score and precision and recall should have also been calculated to show a true balanced performance.

1.4 Improved model

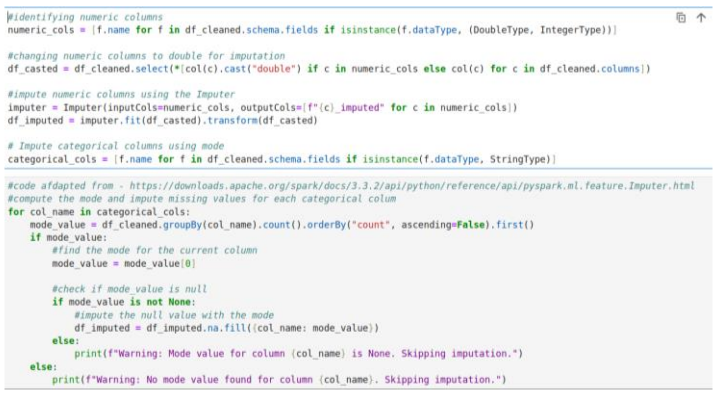
Fig. 1.4-1Handling missing values using Pyspark



```
#replace "?" with null values
df_cleaned = df.replace("?", None)
```

```
#viewing all null value counts - https://www.aporia.com/resources/how-to/count-nan-values-dataframe/
null_counts_imputed = df_cleaned.select([count(when(col(c).isNull(), c)).alias(c) for c in df_cleaned.columns])
null_counts_imputed.show()
```

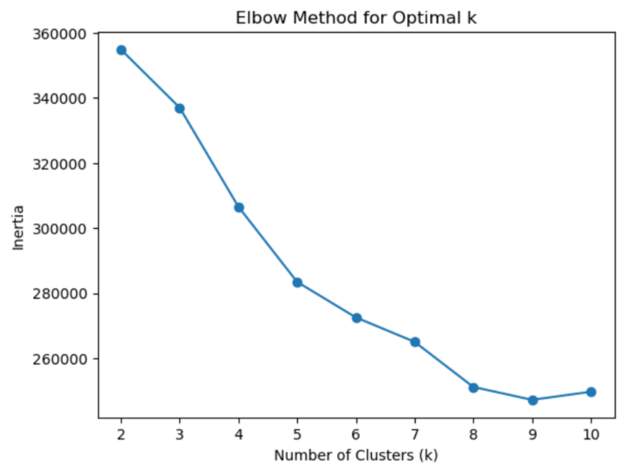
Fig. 1.4-2 imputing the missing values using Pyspark imputer



```python
#identifying numeric columns
numeric_cols = [f.name for f in df_cleaned.schema.fields if isinstance(f.dataType, (DoubleType, IntegerType))]

#changing numeric columns to double for imputation
df_casted = df_cleaned.select(*[col(c).cast("double") if c in numeric_cols else col(c) for c in df_cleaned.columns])

#impute numeric columns using the Imputer
imputer = Imputer(inputCols=numeric_cols, outputCols=[f"{c}_imputed" for c in numeric_cols])
df_imputed = imputer.fit(df_casted).transform(df_casted)

# Impute categorical columns using mode
categorical_cols = [f.name for f in df_cleaned.schema.fields if isinstance(f.dataType, StringType)]
```

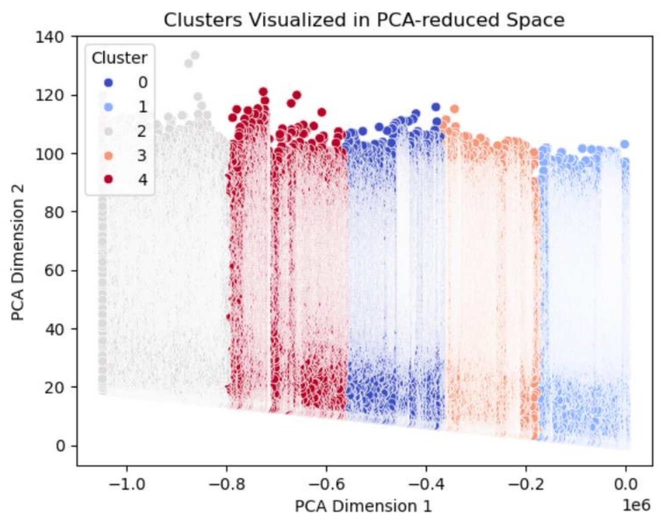
```python
#code afdapted from - https://downloads.apache.org/spark/docs/3.3.2/api/python/reference/api/pyspark.ml.feature.Imputer.html
#compute the mode and impute missing values for each categorical colum
for col_name in categorical_cols:
    mode_value = df_cleaned.groupBy(col_name).count().orderBy("count", ascending=False).first()
    if mode_value:
        #find the mode for the current column
        mode_value = mode_value[0]

        #check if mode_value is null
        if mode_value is not None:
            #impute the null value with the mode
            df_imputed = df_imputed.na.fill({col_name: mode_value})
        else:
            print(f"Warning: Mode value for column {col_name} is None. Skipping imputation.")
    else:
        print(f"Warning: No mode value found for column {col_name}. Skipping imputation.")
```

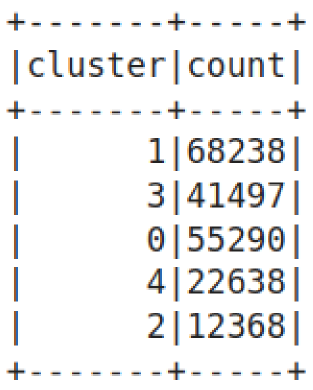
Fig. 1.4-3 Data visualisation using Elbow method to find optimal k



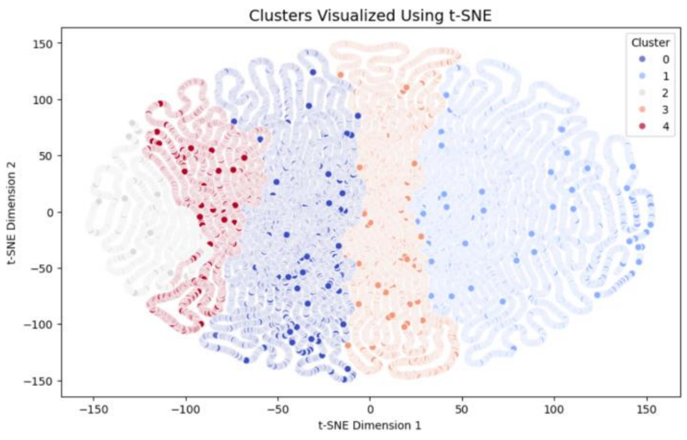
EVALUATION OF RESULTS:

Fig. 1.4-4 Clustering results





```
+-------+-----+
|cluster|count|
+-------+-----+
|      1|68238|
|      3|41497|
|      0|55290|
|      4|22638|
|      2|12368|
+-------+-----+
```



Conclusion: According to the elbow method, the suggested k value is 5. Based on this k value, clustering was done and 5 clusters were found and some variation in data distribution was observed. The silhouette score was also calculated that is 0.7933 that indicates the clusters are distinct. Based on this conclusion, better results were found when the values were imputed instead of deletion and cluster-based classification was performed. We also performed tSNE clustering which gave a similar result hence, verifying our results further. It served as a confirmation tool for clustering and visually verify that the clusters formed by the algorithm are meaningful and distinct. Furthermore, using balanced and whole set of data improved the performance of our prediction model which indicates that class imbalance can mislead the predictions.