Banking Buisness case(data analysis assessment)

Madan K

Problem Statement:

A major bank in Middle East came to NeoStats with help in analysing its current customer base and its marketing campaigns. It wants to understand which customers are most likely to take a term deposit (fixed deposit), and then send this list to their call centre.

Goal of Project:

we have to create classification model for bank so that it makes more easier to them identify the customers who likely to subscribe the Term Deposit(Fixed Deposit)

Flow for analysis and model building:

- **Data Preprocessing**
- **EDA**
- **Visualizations and Reporting**
- **Correlation and Clustering analysis**
- **Feature engineering**
- **Model Building and Evaluation**
- **Conclusion and Insights:**

Dataset:

It contains two dataset where one dataset(45211 rows × 14 columns) consist of customers transaction details and other dataset(45211 rows × 7 columns) details of customer demographic information.

```
df_1
```

	Sno	Customer_number	Insurance	balance	housing	loan	contact	duration	campaign	last_contact_day	previous	poutcome	Term Deposit	Count_Txn
0	0	1001	no	2143	yes	no	NaN	261.0	1	2	0	unknown	no	351.0
1	1	1002	no	29	yes	no	unknown	151.0	1	2	0	unknown	no	326.0
2	2	1003	no	2	yes	yes	unknown	76.0	1	2	0	NaN	no	422.0
3	3	1004	no	1506	yes	no	unknown	92.0	1	2	0	unknown	no	113.0
4	4	1005	no	1	no	no	unknown	198.0	1	2	0	unknown	no	342.0

```
df_2
```

	Customer_number	age	job	marital	education	Annual Income	Gender
0	1001	58	management	married	tertiary	839368	M
1	1002	44	technician	single	secondary	1452858	M
2	1003	33	entrepreneur	married	NaN	4851383	F
3	1004	47	blue collar	married	unknown	3207754	F
4	1005	33	unknown	single	unknown	2562322	M

Preprocessing Transaction Details Data (df_1):

```
null_df1=(df_1.isnull().sum()/len(df_1))*100
null_df1
```

```
Sno                0.000000
Customer_number    0.000000
Insurance          0.000000
balance            0.121652
housing            0.000000
loan               0.017695
contact            0.095110
duration           0.086262
campaign           0.000000
last_contact_day   0.000000
previous           0.000000
poutcome           0.033178
Term Deposit       0.017695
Count_Txn          0.002212
dtype: float64
```

```
null features:['balance', 'loan', 'contact', 'duration', 'poutcome', 'Term Deposit', 'Count_Txn']
not_null features:['Sno', 'Customer_number', 'Insurance', 'housing', 'campaign', 'last_contact_day', 'previous']
```

- As we can see very small percentage of null values in features balance,loan,contact,duration,poutcome,Term Deposit,Count_Txn

• Check For Duplicates:
•There are no duplicate rows in df_1 dataset
• will carry out the null value imputation based on their distribution for numerical features and replace with mode in categorical feature

Preprocessing Transaction Details Data (df_2):

Missing values in df_2 Dataset

```
null_df2=(df_2.isnull().sum()/len(df_2))*100
null_df2
```

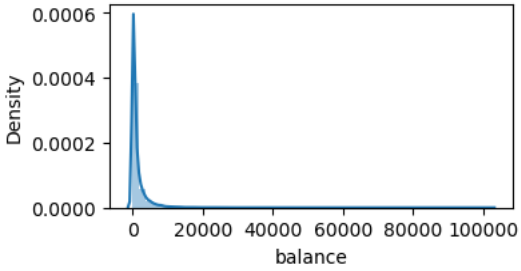
```
Customer_number     0.000000
age                 0.000000
job                 0.028754
marital             0.039813
education           0.046449
Annual Income       0.037601
Gender              0.000000
dtype: float64
```

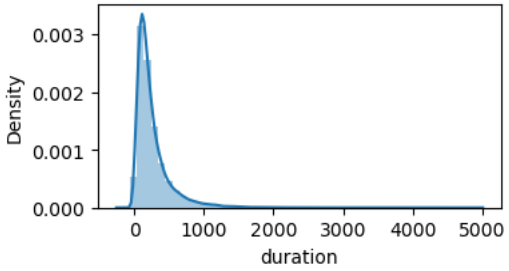
```
null features:['job', 'marital', 'education', 'Annual Income']
not_null features:['Customer_number', 'age', 'Gender']
```

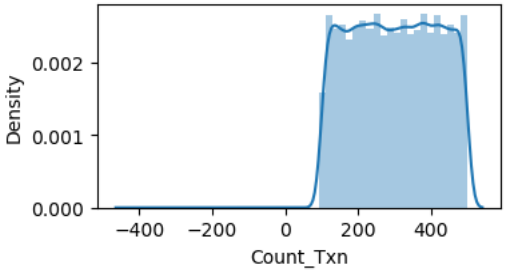
- As we can see very small percentage of null values in features job,marital,education,Annual Income features

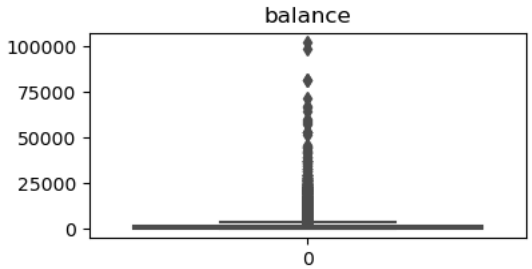
• Check For Duplicates:
•There are no duplicate rows in df_2 dataset
•will carry out the null value imputation based on their distribution for numerical features and replace with mode in categorical feature

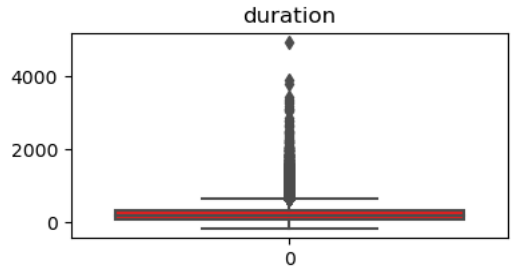
Null imputation based on distribution of Numerical Features(dataset 1):

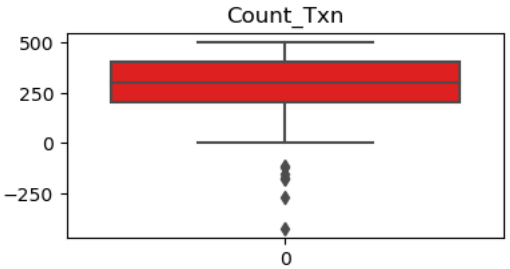












Replace the null values with median

```
df_1['balance'].fillna(df_1['balance'].median(), inplace=True)
df_1['duration'].fillna(df_1['duration'].median(), inplace=True)
df_1['Count_Txn'].fillna(df_1['Count_Txn'].median(), inplace=True)
```

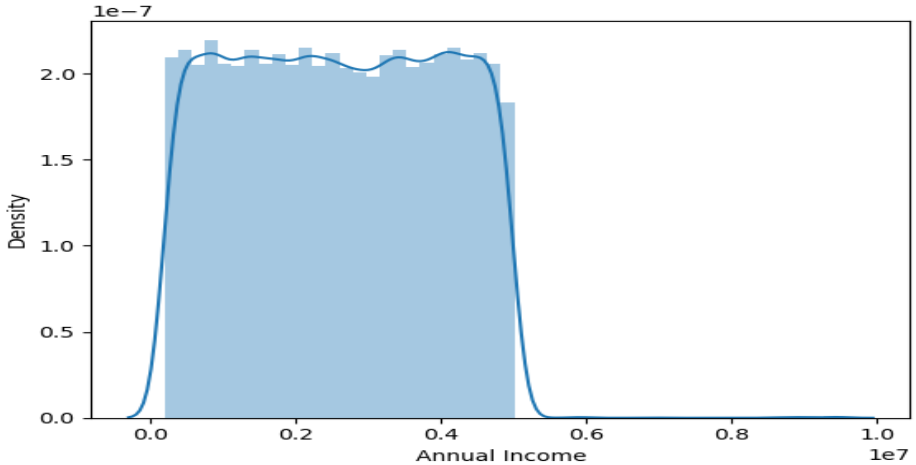
Replace the null values in categorical values with mode

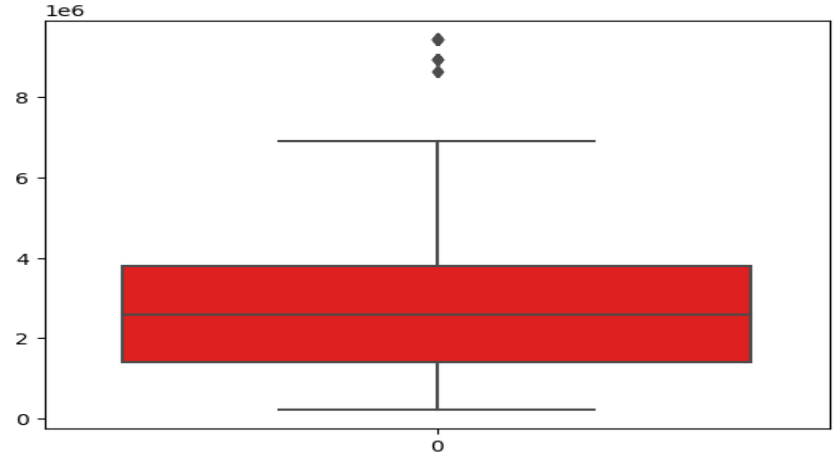
```
categorical_columns = ['loan', 'contact', 'poutcome', 'Term Deposit']

for column in categorical_columns:
    mode_value = df_1[column].mode()[0]
    df_1[column].fillna(mode_value, inplace=True)
```

• Features such as balance,duration are rightly skwed and Count_Txn are negatively skewed so its better replace the null values by median

•Skewed features such as 'balance', 'duration', and 'Count_Txn' are better to be imputed with the median rather than the mean. The median is less sensitive to outliers and provides a better representation of the central tendency in skewed distributions.

Null imputation based on distribution of Numerical Features(dataset 2):





Annual income feature is right skewed we replace with median value and got few outliers

```
: df_2['Annual Income'].fillna(df_2['Annual Income'].median(),inplace=True)
```

```
df_2['marital'].fillna(df_2['marital'].mode()[0], inplace=True)
df_2['education'].fillna(df_2['education'].mode()[0], inplace=True)
df_2['job'].fillna(df_2['job'].mode()[0], inplace=True)
```

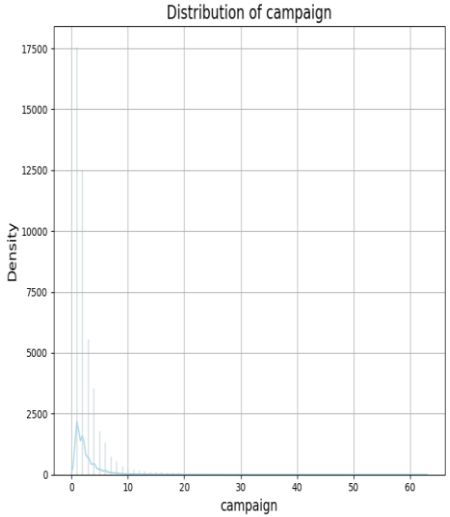
Merge Dataset 1 & 2:

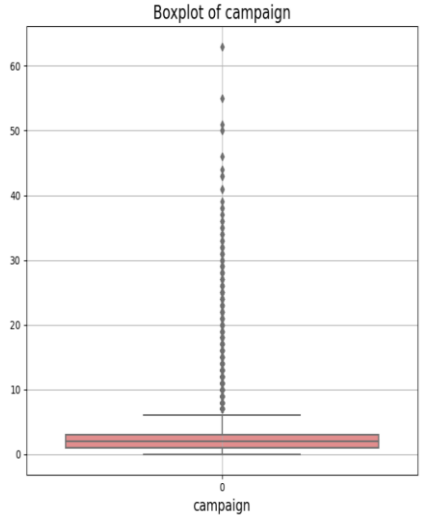
- Data preprocessing for both datasets for df_1 and df_2 are done and now we merge them as one dataset and perform analysis
- Given that we have a common 'customer_no' column in both datasets, using an inner join would likely be the best choice in most cases and allows us to do analyzing only the data for customers who exist in both datasets, ensuring that you're working with complete information for those customers.

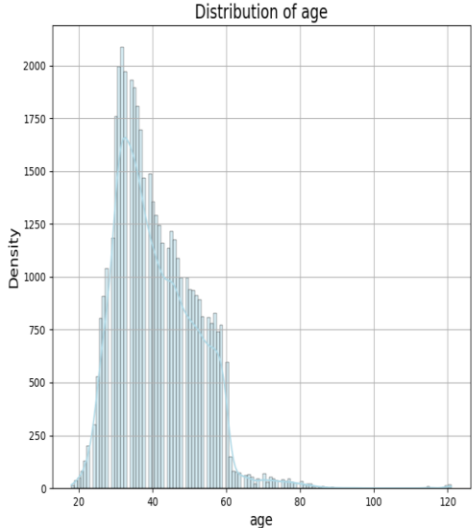
```
merged_df.columns
```

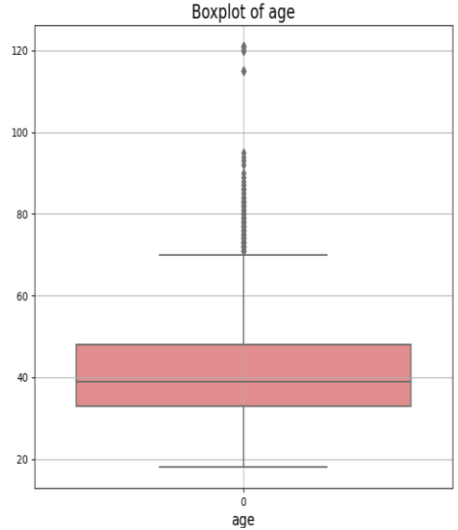
```
Index(['Sno', 'Customer_number', 'Insurance', 'balance', 'housing', 'loan',
       'contact', 'duration', 'campaign', 'last_contact_day', 'previous',
       'poutcome', 'Term Deposit', 'Count_Txn', 'age', 'job', 'marital',
       'education', 'Annual Income', 'Gender'],
      dtype='object')
```

Outlier detection:







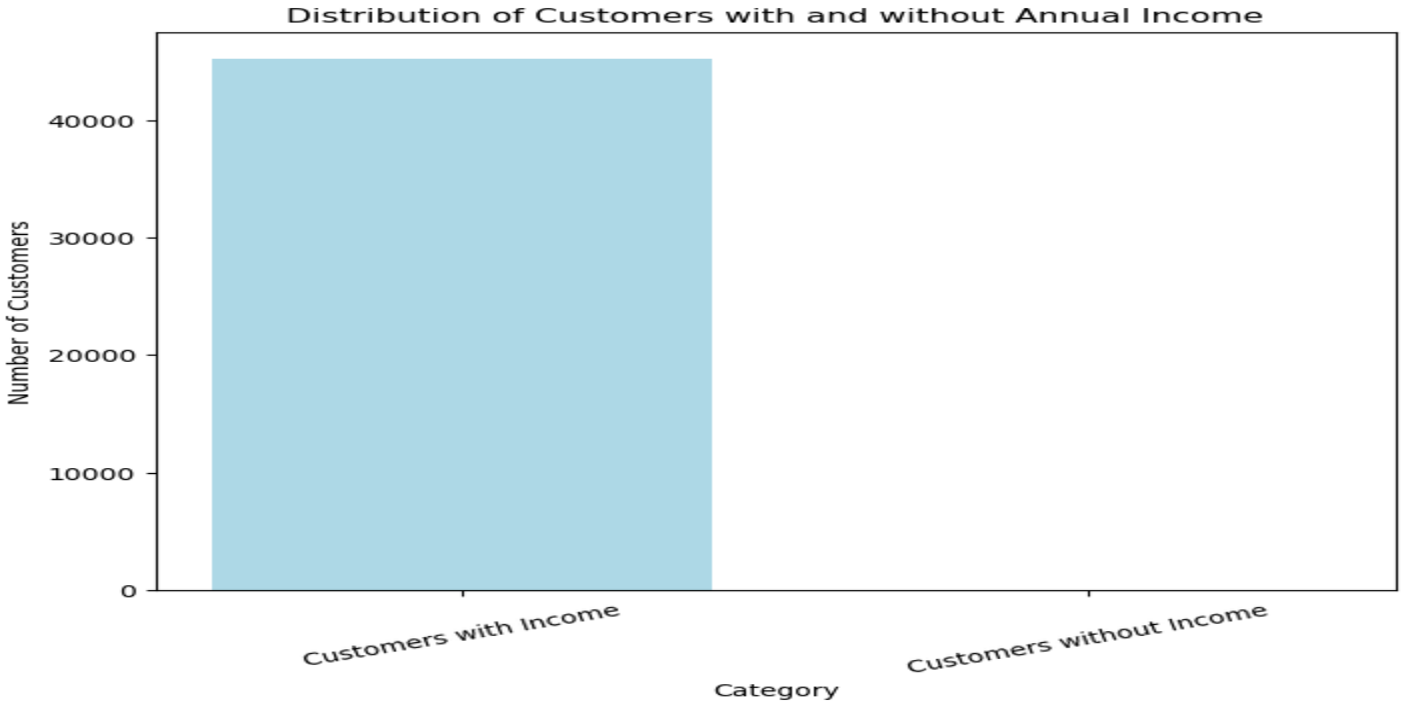


- Almost all the numerical features contain the outliers and skewed,
- So before treating the outliers we have to build base model,so accordingly we treat them if needed to improve the model performance

Data Analysis & Visualization:

1. Income Insights:

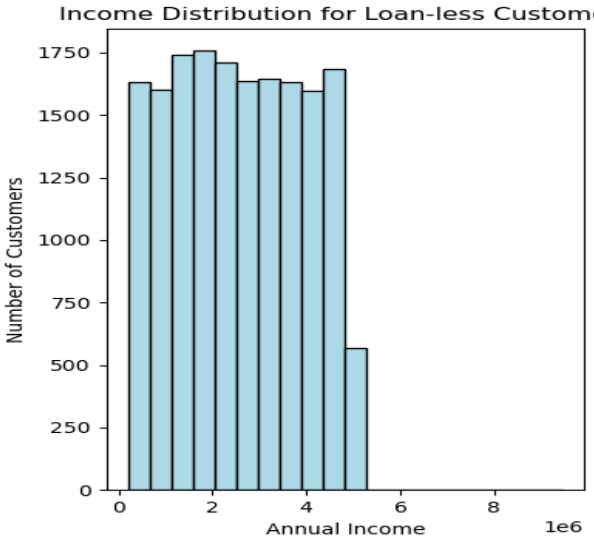
• How many customers have no annual income? Plot and present the data distribution of these customers.

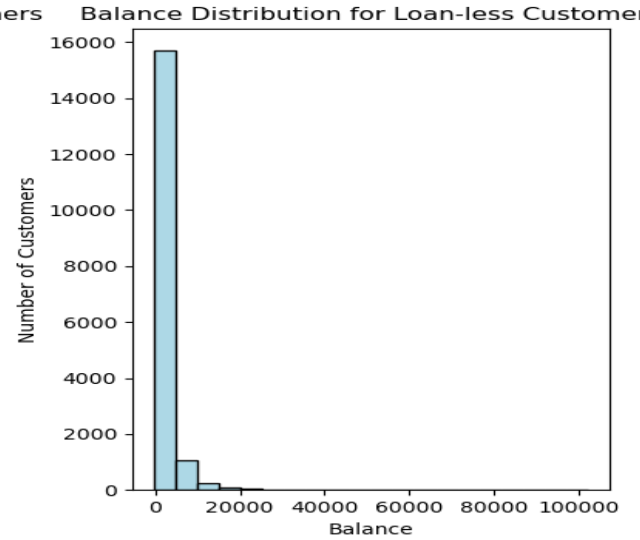


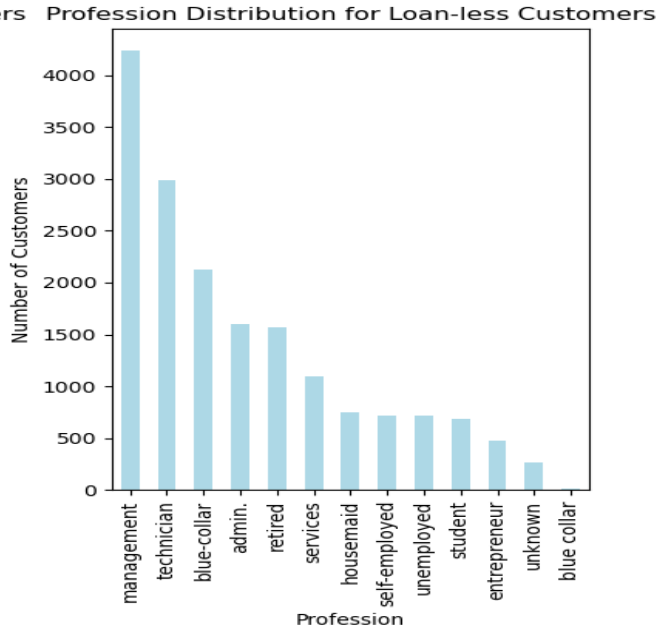
Number of customers with no annual income: 0

2) Loan-less Customers Profile:

• Filter out customers who don't have any type of loan. Plot the distribution of their Income, balance, and profession. How do these metrics differ from those with loans?







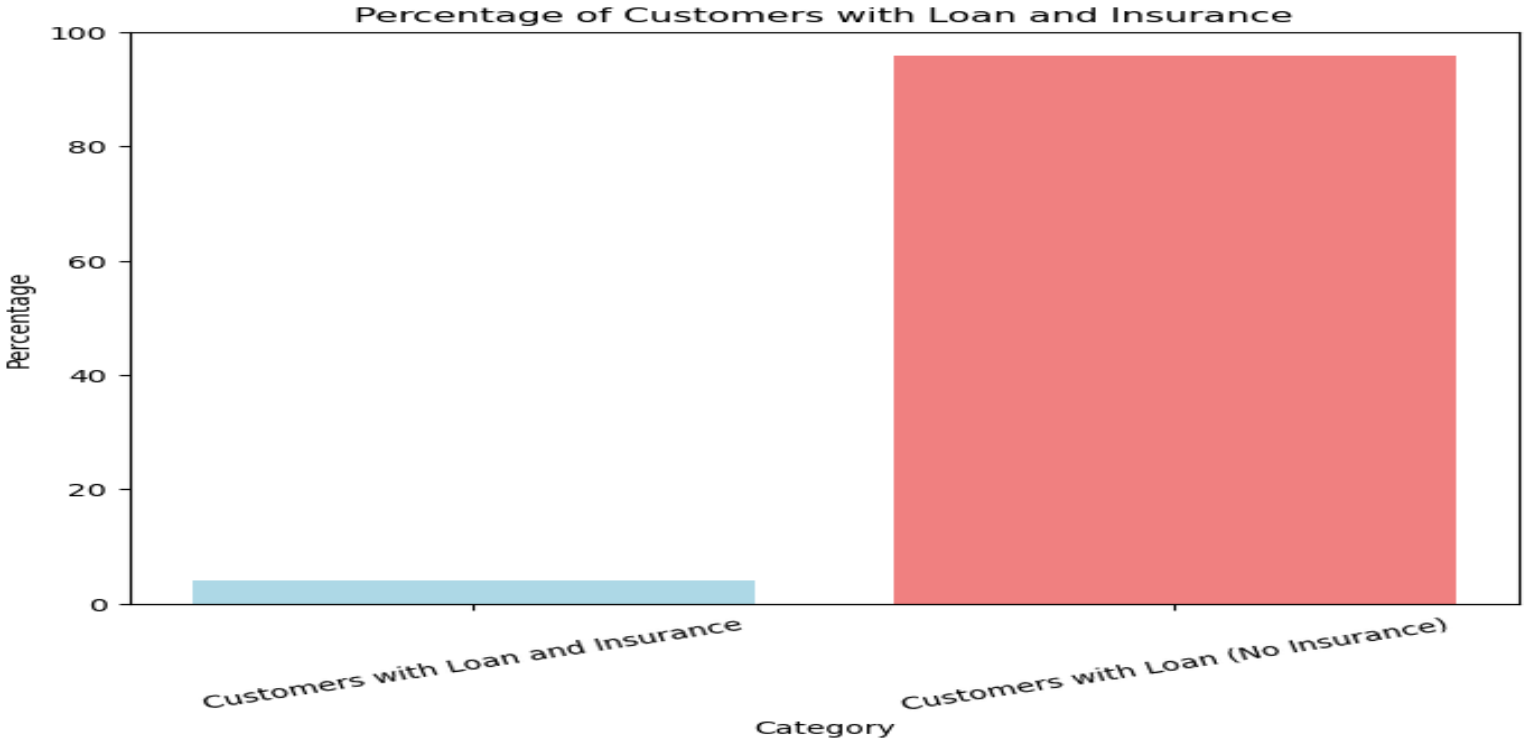
•**Annual Income**: The mean annual income is slightly higher for loaned customers (2,602,288) compared to loan-less customers (2,587,248). However, the difference in means is relatively small.

•**Balance**: Loaned customers have a lower mean balance (1,208.27) compared to loan-less customers (1,752.97). Loan-less customers also have a wider range of balance values, with both negative and high balances.

•**job**:also People dont take loans are from management,technicians etc and customers who take loans are having blue collar jobs

3. Loan and Insurance Analysis:

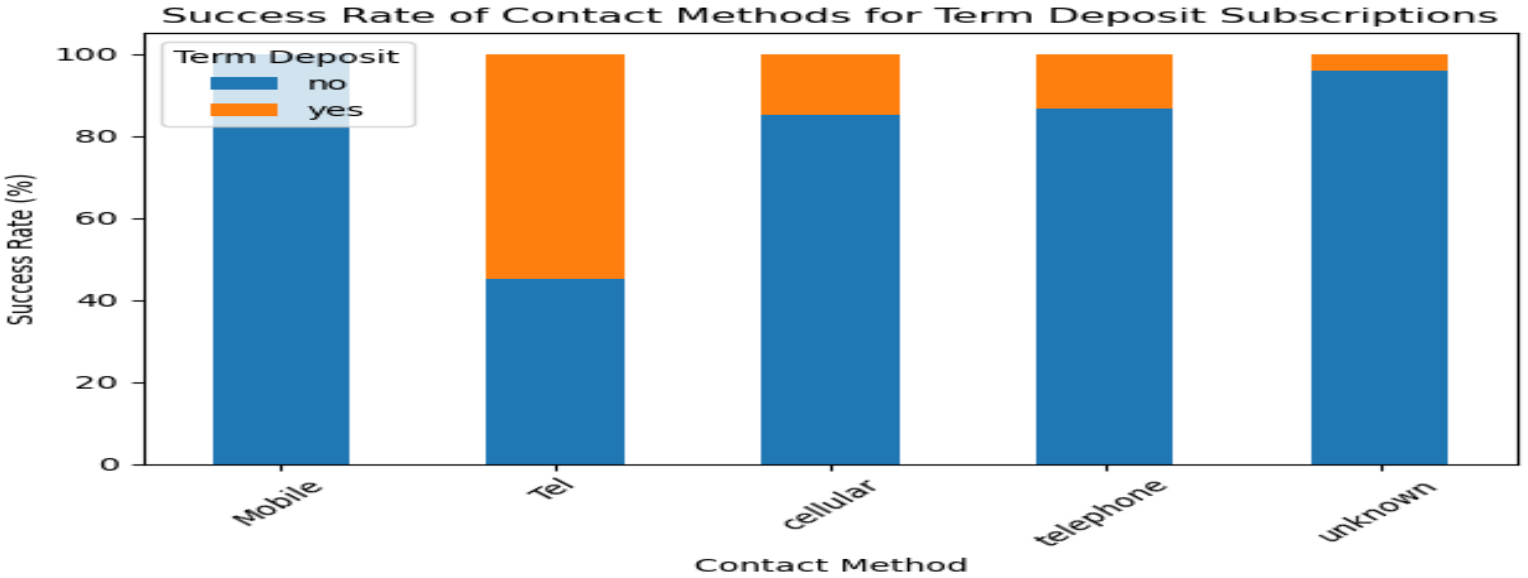
• Calculate the percentage of customers with a loan who have taken out insurance. Visualize this data and discuss potential implications.



Percentage of customers with a loan who have insurance: 4.16%

4)Communication Strategy Insights:

Analyse and summarize the best Contact method (with the highest success percentage) to contact people to ascertain the status of term deposit subscriptions.

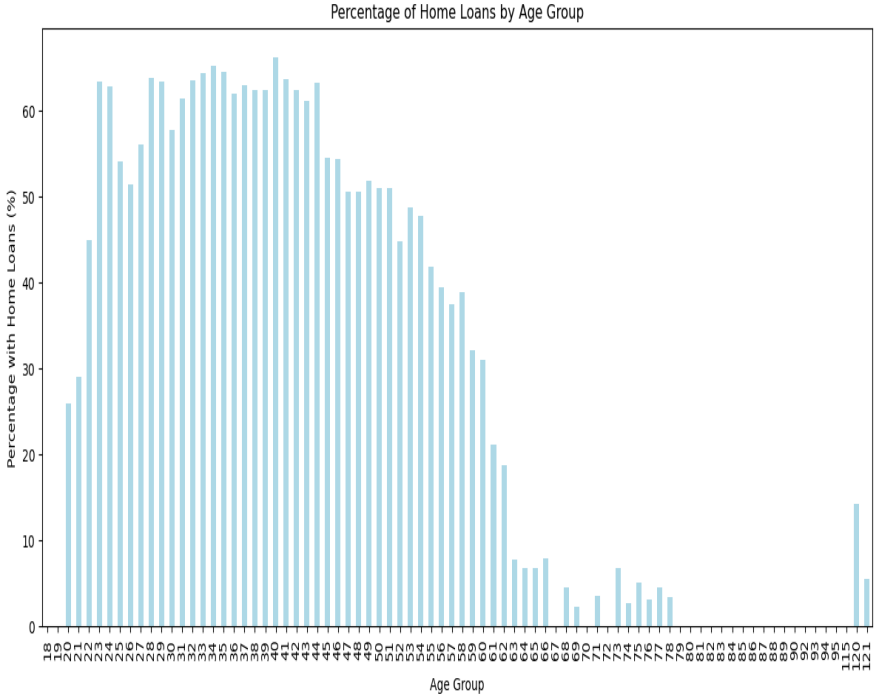


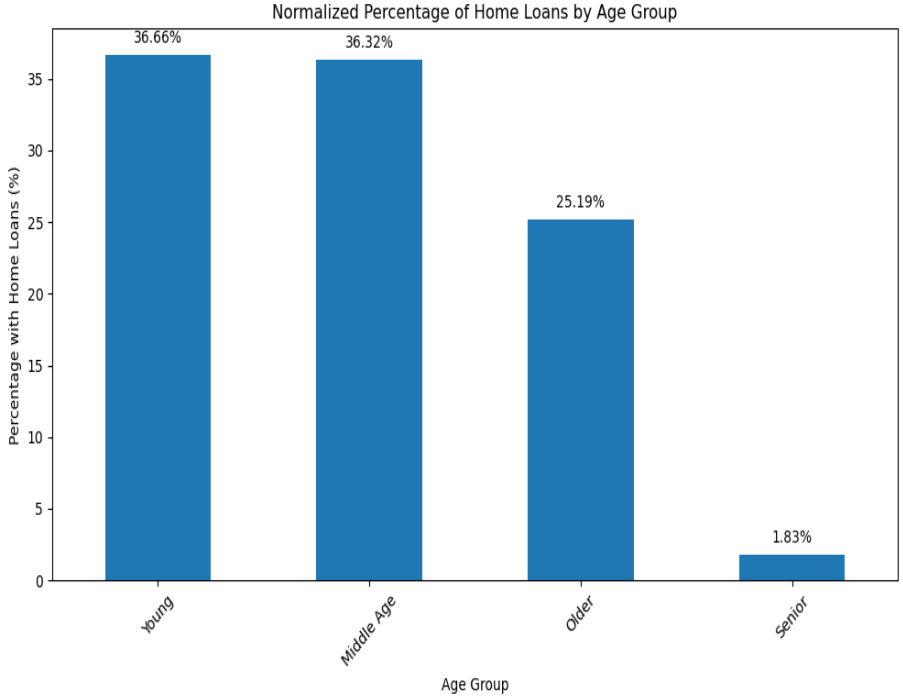
•It seems that "telephone" and "Tel" are likely two variations of the same contact method, but with slightly different labels might be due to data inconsistency .

•We can assume that these both are same contact method,when the customers are contacted through Telephone there are more chances that they would subscribe to Fixed deposit

5)Age and Home Loans:

Determine which age group has the highest percentage of home loans. Present this data visually and discuss possible reasons.

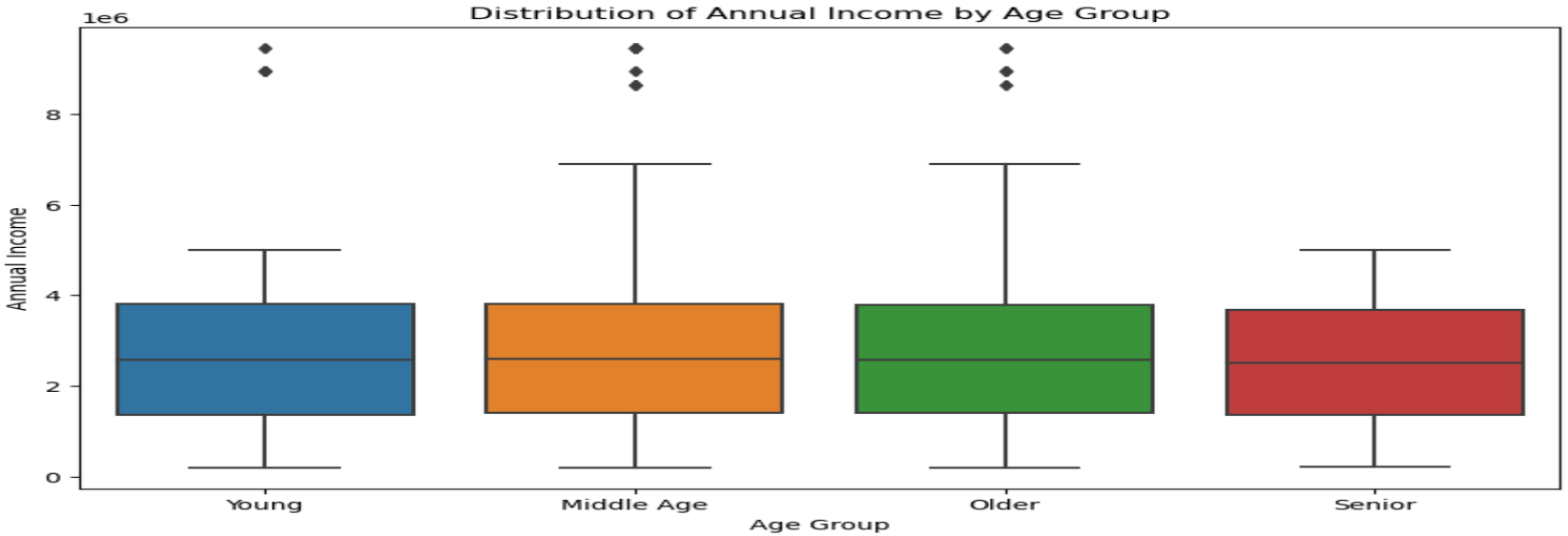




- The age group with the highest percentage of home loans is 40 with a percentage of 66.27%.
- Customers aging between 18 to 35 has taken most home loans with 36.66% which is approximately equal to customers who have taken loan aging 35-50

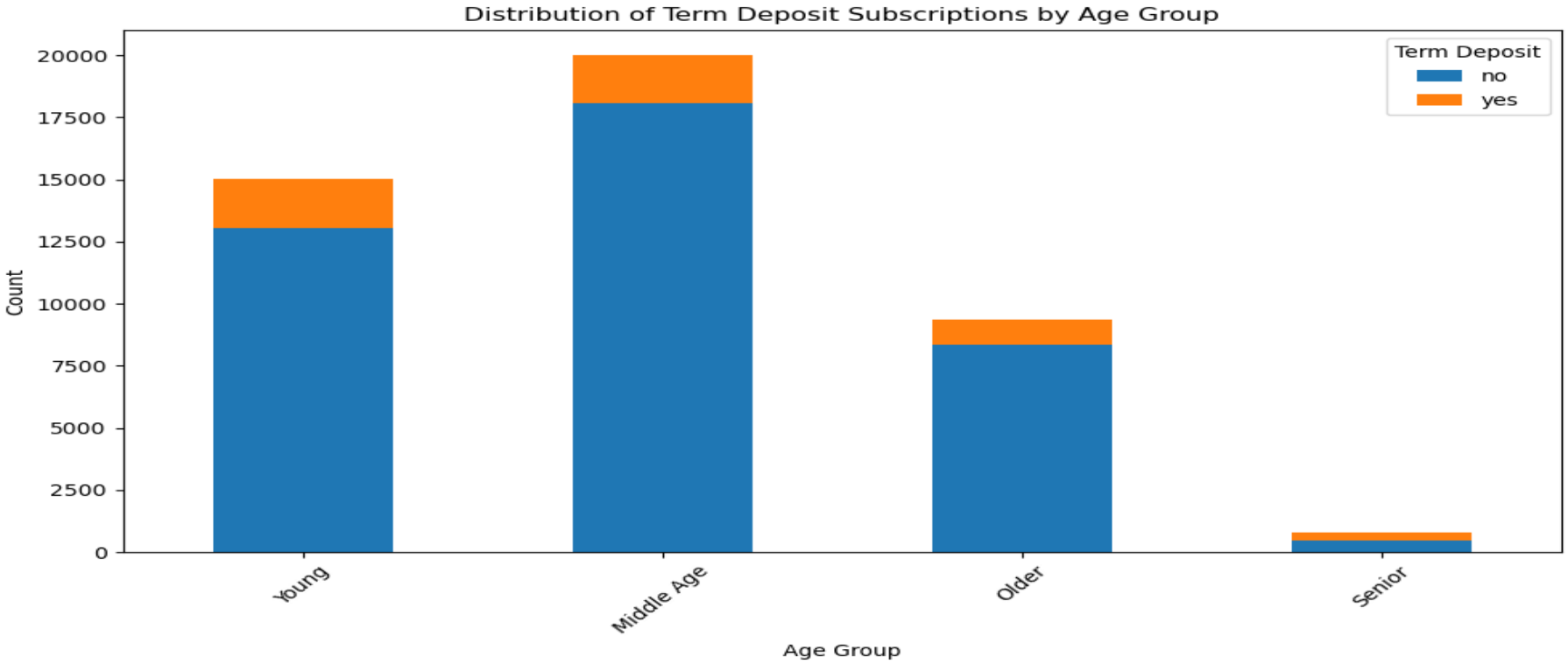
6) Income and Age Relationship:

Investigate any relationships between annual income and age group. Use appropriate plots and statistics to present the findings.



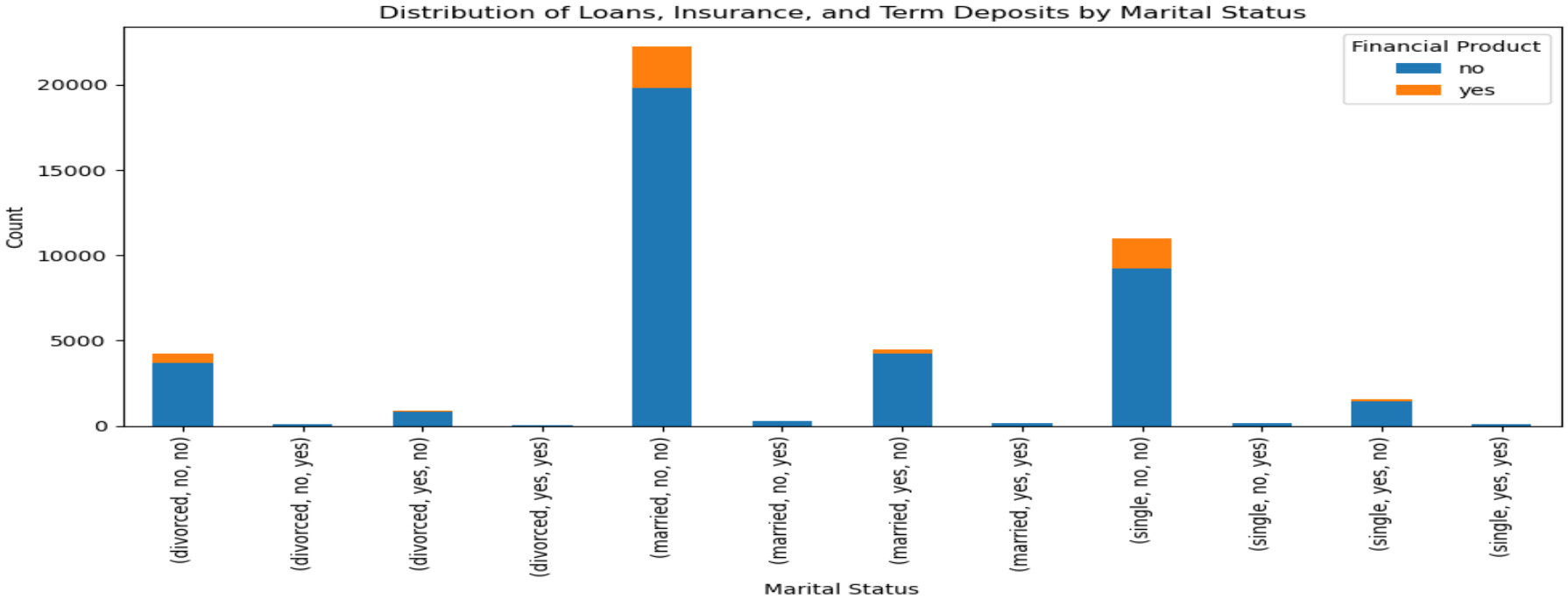
- Middle aged and older aged customers have approximately same annual income
- But across the age groups, we found that the average annual income values were relatively similar. There is not a substantial variation in average income among these groups.
- While we observed similar average income values across age groups, this does not shows relationship between the variables.

Other Analysis:



• Customers aging between 18 to 50 has subscribed to Term deposit compare to customers aged above the 50

Other Analysis:

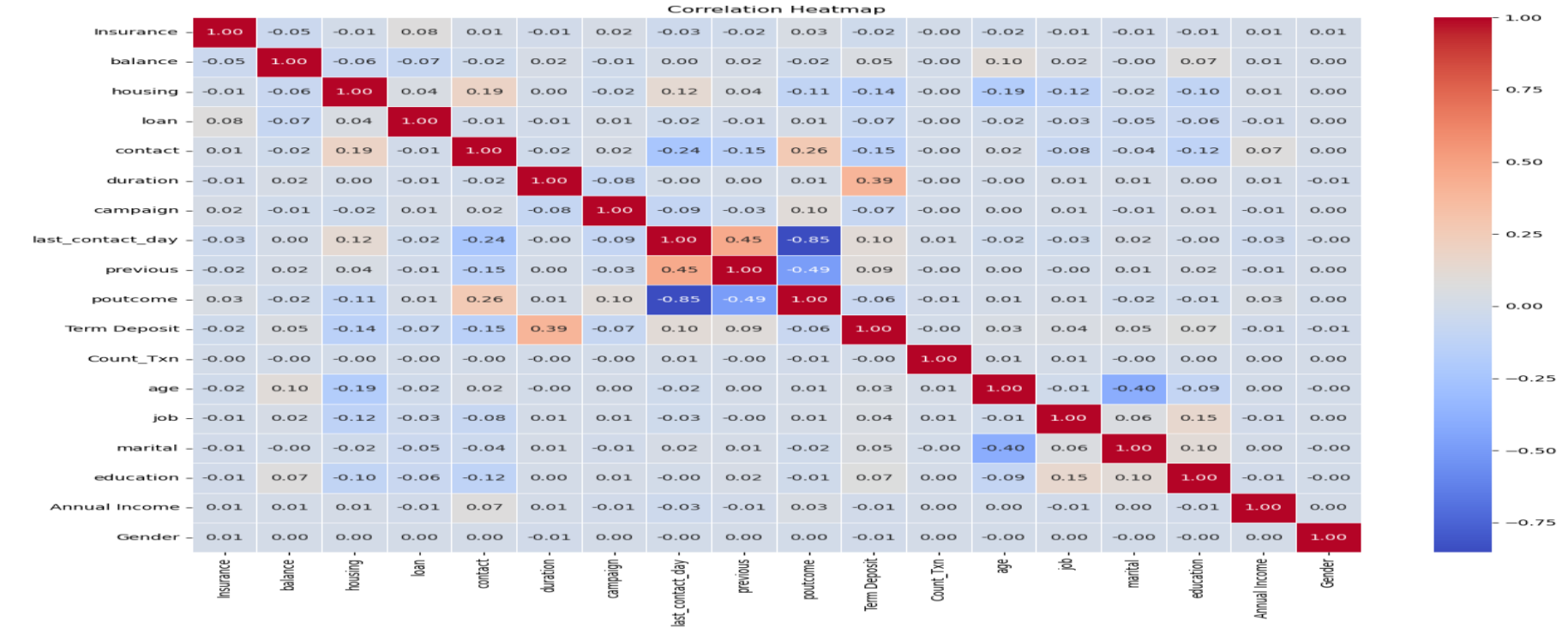


•**Divorced Individuals**: Divorced individuals who do not have a loan show a relatively higher chances of subscribing term deposit compared to those with a loan.

•**Single Individuals**: The success rates among single individuals are relatively consistent, regardless of loan status. This indicates that for single individuals, the presence of a loan may not significantly impact their likelihood of subscribing to a term deposit.

•**Married Individuals:** Those without a loan have a higher success rate compared to those with a loan.

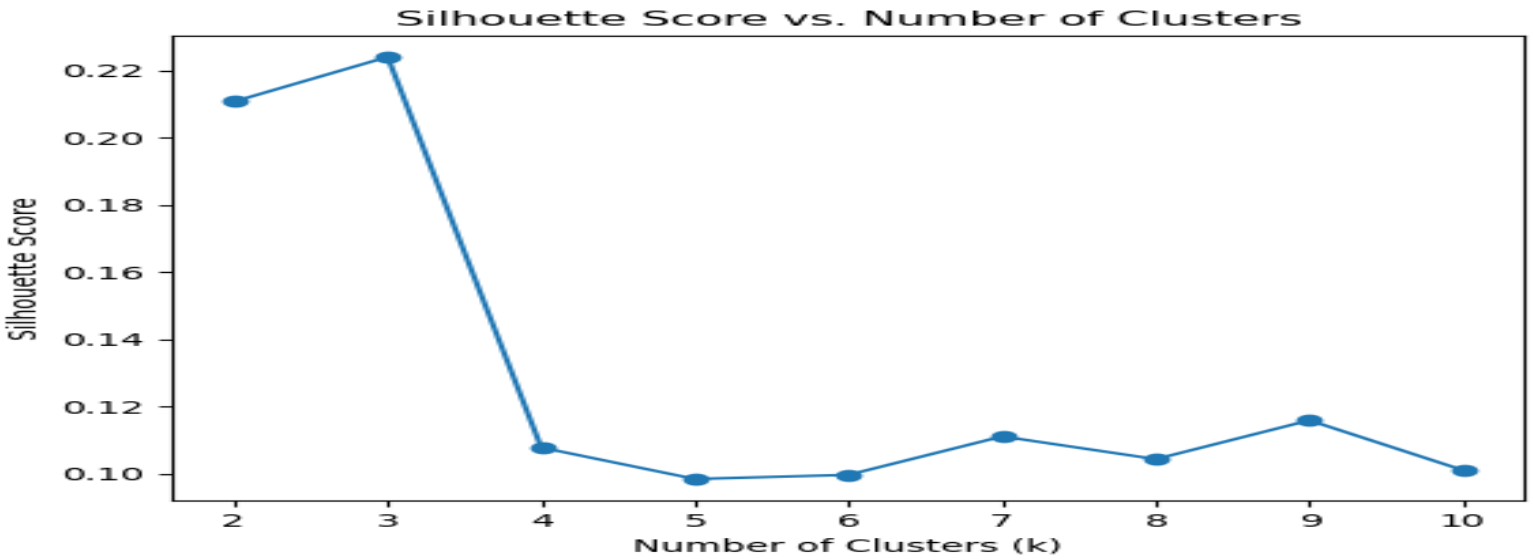
Correlation Analysis:



• Out of all features Insurance, Balance, Housing, Loan, Contact, Duration, Campaign, Last Contact Day, Previous, Poutcome,the highest correlation appears to be with the "Duration" column, suggesting that the duration of the contact has a relatively strong positive correlation with the success of the term deposit subscription. This is intuitive, as longer conversations might indicate more interest and engagement.

•This column appears to represent the count of transactions and its correlation with the "Term Deposit" column is close to zero, indicating that it has little influence on term deposit subscription.

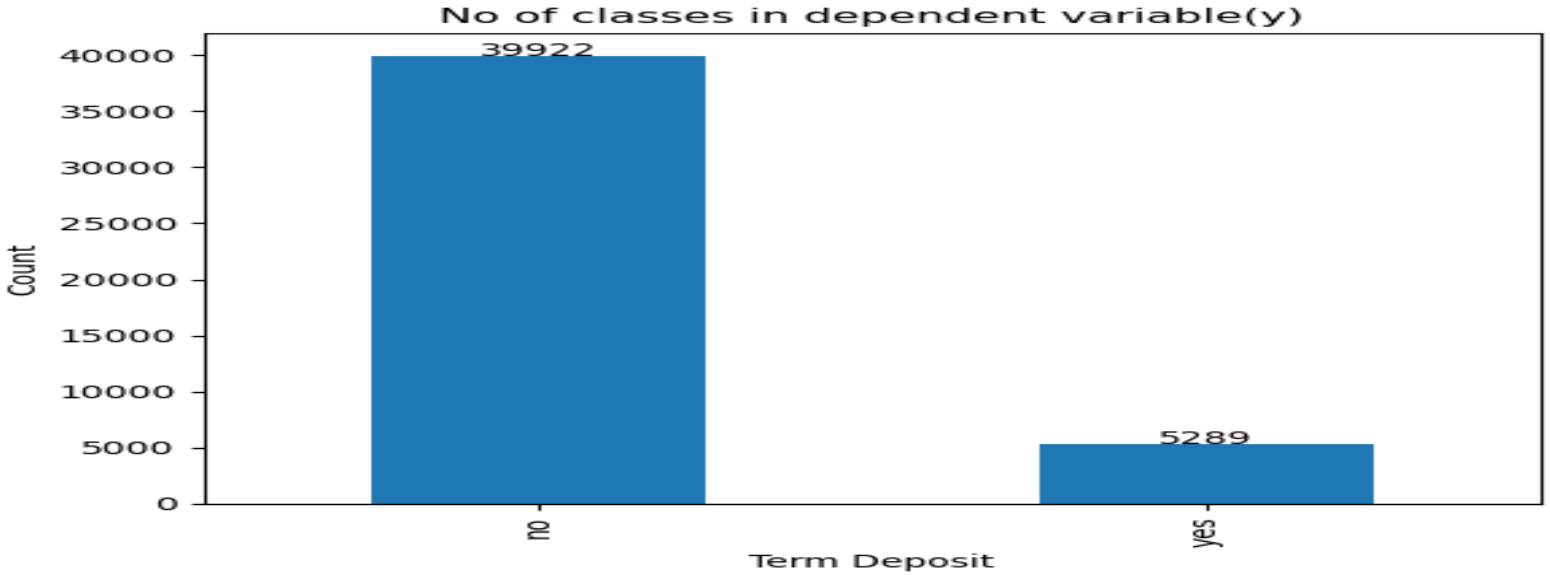
Clustering Analysis:



• A silhouette score of 0.1 to 0.22 suggests that the clusters are overlapping and not well-separated so clusters are overlapping

• Since earlier we had performed the correlation analysis, in that we have seen that there was no relationship among the variables so these might be the reason for overlapping of clusters,since Clustering algorithms attempt to group similar data points together based on certain features

Check for imbalanced dataset:



- The output class is imbalanced in our dataset. The output class has two categories: "Yes" and "No" The count of samples in each category is as follows:
- "no": 39922 samples
- "yes": 5289 samples
- Since our dataset is imbalanced so lets create the base model ans check for model accuracy further we carry out feature selection,upsampling the data

Model Building:

Logistic Regression

```
log_clf = LogisticRegression()
log_clf.fit(X_train, y_train)
yhat = log_clf.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, log_clf.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 88.73%
Training Accuracy: 89.29%
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      7952
           1       0.59      0.22      0.32      1091

    accuracy                           0.89      9043
   macro avg       0.75      0.60      0.63      9043
weighted avg       0.86      0.89      0.86      9043
```

Support Vector Machines

```
svm_clf = SVC(kernel='rbf')
svm_clf.fit(X_train, y_train)
yhat = svm_clf.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, svm_clf.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 89.53%
Training Accuracy: 90.72%
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      7952
           1       0.65      0.28      0.39      1091

    accuracy                           0.90      9043
   macro avg       0.78      0.63      0.67      9043
weighted avg       0.88      0.90      0.88      9043
```

Results:

•As we can see that for each model built we can observe that

•Precision:class 0(no)(negative class) has a precision higher, and class 1 (positive class) has a precision is low . This suggests that the model is better at correctly predicting class 0.

•Recall: Class 0 has a recall high, indicating that the model is good at identifying true negatives. Class 1 has a recall is low comparitively, suggesting that the model struggles with correctly identifying true positives for class 1.

•Since the data is imbalanced we could upsample the data

Model Building after upsampling using smote method:

Support Vector Machines

```
svm_clf1 = SVC(kernel='rbf')
svm_clf1.fit(X_train, y_train)
yhat = svm_clf1.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, svm_clf1.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 85.63%
Training Accuracy: 86.94%
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.81      0.85      7908
           1       0.83      0.90      0.86      8061

    accuracy                           0.86     15969
   macro avg       0.86      0.86      0.86     15969
weighted avg       0.86      0.86      0.86     15969
```

Random Forest

```
rf_clf1 = RandomForestClassifier(n_estimators=500, random_state=42
rf_clf1.fit(X_train, y_train)
yhat = rf_clf1.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, rf_clf1.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 91.30%
Training Accuracy: 95.05%
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.89      0.91      7908
           1       0.90      0.94      0.92      8061

    accuracy                           0.91     15969
   macro avg       0.91      0.91      0.91     15969
weighted avg       0.91      0.91      0.91     15969
```

Results:

- The model accuracy has been improved where precision and recall for class 1(yes) are well improved
- We got good result by using algorithm such as xgboost,knn,random forest after upsampling

Feature selection using chi2:

```
X_train_selected = X_train[chi2_selected_features]
X_test_selected = X_test[chi2_selected_features]
print("Top 10 Selected Features:")
print(chi2_selected_features)
```

```
Top 10 Selected Features:
Index(['Insurance', 'housing', 'loan', 'contact', 'duration', 'campaign',
       'last_contact_day', 'poutcome', 'marital', 'education'],
      dtype='object')
```

Support vector machines

```
svm_clf2 = SVC(kernel='rbf')
svm_clf2.fit(X_train, y_train)
yhat = svm_clf2.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, svm_clf2.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 82.90%
Training Accuracy: 83.33%
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.80      0.82      7908
           1       0.81      0.86      0.84      8061

    accuracy                           0.83     15969
   macro avg       0.83      0.83      0.83     15969
weighted avg       0.83      0.83      0.83     15969
```

Random forest

```
rf_clf2 = RandomForestClassifier(n_estimators=500, random_state=42,
rf_clf2.fit(X_train, y_train)
yhat = rf_clf2.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, rf_clf2.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 87.75%
Training Accuracy: 90.25%
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.85      0.87      7908
           1       0.86      0.91      0.88      8061

    accuracy                           0.88     15969
   macro avg       0.88      0.88      0.88     15969
weighted avg       0.88      0.88      0.88     15969
```

Feature selection using RFE:

```
print("Top 10 Selected Features:")
print(rfe_selected_features)
```

```
Top 10 Selected Features:
Index(['Insurance', 'housing', 'loan', 'contact', 'campaign', 'previous',
       'poutcome', 'marital', 'education', 'Gender'],
      dtype='object')
```

```
log_clf3 = LogisticRegression()
log_clf3.fit(X_train, y_train)
yhat = log_clf3.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, log_clf3.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 66.03%
Training Accuracy: 66.62%
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.63      0.65      7908
           1       0.65      0.69      0.67      8061

    accuracy                           0.66     15969
   macro avg       0.66      0.66      0.66     15969
weighted avg       0.66      0.66      0.66     15969
```

Analysis before and after feature selection:

- Test and Training Accuracy(Random Forest):
- Before Feature Selection: The initial test accuracy was 91.30%, and the training accuracy was 95.05%.
- After Feature Selection: The test accuracy dropped to 87.75%, and the training accuracy dropped to 90.25%.
- The initial model had a higher test accuracy compared to the model after feature selection, suggesting that the original model was performing better on unseen data.
- The drop in training accuracy after feature selection indicates that the model is now less overfit to the training data, which can be a positive outcome.
- feature selection seems to have had a mixed impact on the model's performance.

UsingTree based for feature selection:

```
feature_importances = rf_clf.feature_importances_
feature_importance_list = list(zip(df.columns, feature_importances))
feature_importance_list.sort(key=lambda x: x[1], reverse=True)
top_features = [feature for feature, importance in feature_importance_list[:10]]
```

```
top_features
```

```
['duration',
 'campaign',
 'balance',
 'Count_Txn',
 'contact',
 'education',
 'Term Deposit',
 'age',
 'poutcome',
 'housing']
```

support vector machines

```
svm_clf4 = SVC(kernel='rbf')
svm_clf4.fit(X_train, y_train)
yhat = svm_clf4.predict(X_test)
accuracy = accuracy_score(y_test, yhat)
print('Test Accuracy: %.2f%%' % (accuracy * 100))
train_accuracy = accuracy_score(y_train, svm_clf4.predict(X_train))
print('Training Accuracy: %.2f%%' % (train_accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, yhat))
```

```
Test Accuracy: 100.00%
Training Accuracy: 100.00%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7908
           1       1.00      1.00      1.00      8061

    accuracy                           1.00     15969
   macro avg       1.00      1.00      1.00     15969
weighted avg       1.00      1.00      1.00     15969
```

```
cv_scores = cross_val_score(svm_clf4, X_train, y_train, cv=5, scoring='accuracy')
print("Cross-Validation Scores:", cv_scores)
print("Average Cross-Validation Score: %.2f%%" % (cv_scores.mean() * 100))
```

```
Cross-Validation Scores: [1. 1. 1. 1. 1.]
Average Cross-Validation Score: 100.00%
```

•Both feature selection methods such as chi2 and tree based methods gave good accuracy on both Train and test data with well balanced precision and recall for class 1(Yes) and drop in overfitting.

Conclusion:

•The common features in both the Random Forest Feature Selection and the Chi-Squared Test results are:
•Out of all features in data only these features tends to be more significant to Term Deposit:
•'contact'
•'duration'
•'campaign'
•'poutcome'
•'housing'
•**Contact:** This feature could be crucial as it indicates the communication method used to reach out to customers. A more effective and personalized contact method could positively influence a customer's decision to subscribe.
•**Duration:** The duration of the previous contact with the customer is likely to be significant. Longer interactions might imply more engagement and interest, increasing the chance of a term deposit subscription.
•**Campaign:** The number of contacts made during the campaign could play a role. Too many contacts might lead to annoyance, while an optimal number could improve subscription rates.
•**Poutcome:** The outcome of the previous marketing campaign (poutcome) suggests that if a customer has shown interest or subscribed before, they might be more likely to subscribe again in the current campaign.
•**Housing:** This feature relates to whether a customer has a housing loan. Customers with more stable financial situations (without housing loans) might be more inclined to invest in a term deposit.

Best model after Upsampling:

•K-Nearest Neighbors (KNN):
•Test Accuracy: 91%
•Training Accuracy: 93%
•KNN performed on par with Random Forest,xgboost achieving 91% accuracy on the test set. It also showed consistent performance between training and test data

Best model after Feature Selection using chi2:

•Xgboost:
•Test Accuracy: 90%
•Training Accuracy: 88%
•Xgboost performed on par with Random Forest,svm,knn,xgboost achieving 88% accuracy on the test set. It also showed consistent performance between training and test data

Thank You

Contact details:
Ph.no:+91 8660419121
email:madan.maddy890@gmail.com