Week 13 Deliverables

Name: Riwaj Neupane
Email: neupaneriwaj64@gmail.com
Country: Nepal
College/Company: N/A
Specialization: Data Science

Problem Description:

ABC Bank wants to sell its term deposit product to customers and before launching the product they want to develop a model which helps them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

Data Description:

The dataset going to be used for the analysis is called “bank-additional-full.csv”, which contains 41188 observations and 21 features, encompassing features related to clients’ basic information such as age, job, marital status, education, credit in default, housing, and loan; details about contact such as contact communication type, last contact month, last contact day, last contact duration, number of contacts, etc., and information about marketing campaigns like outcome, employment variation rate, consumer price index, consumer confidence index, euribor 3 month rate, and number of employees. We also have the target variable y, which is the answer for the yes-no question “has the client subscribed a term deposit?”, and it will be used in future prediction.

Feature Name	Type	Data Type	Number of unknowns	Number of Outliers	Comments
age	Numerical	int	0	381	Replace with upper bound defined as Q3+IQR
job	Categorical	str	330	0	Replace with mode
martial	Categorical	str	80	0	Replace with mode
education	Categorical	str	1731	0	
default	Categorical	str	8597	0	Leave unknown as its own type
housing	Categorical	str	990	0	Replace with mode
loan	Categorical	str	990	0	Replace with mode
contact	Categorical	str	0	0	
month	Categorical	str	0	0	
day_of_week	Categorical	str	0	0	
duration	Numerical	int	0	861	Replace with upper bound defined as Q3+IQR
campaign	Numerical	int	0	0	
previous	Numerical	int	0	0	
poutcome	Categorical	str	0	0	
emp.var.rate	Numerical	float64	0	0	
cons.price.idx	Numerical	float64	0	0	
cons.conf.idx	Numerical	float64	0	0	
euribor3m	Numerical	float64	0	0	
nr.employed	Numerical	float64	0	0	

y	Categorical	str	0	0	

Problems in the Data (number of NA values, outliers , skewed etc):

There are 6 categorical features with missing data (job, education, marital, default, housing, & loan). There is one numerical feature ("duration") that contains outlier data. Specifically, we have the mean for "duration" is around 258, but the maximum value is 4918, which indicates the existence of outliers. And in general, the dataset is imbalanced, as the target variable for the predictive classification model skews highly to the "N" case.

Approaches to Overcome These Problems:

For categorical data feature with unknown as category for (job, education, marital, default, housing, & loan). Replacing unknown we can use 2 method:

1. Replacing with mode:

   Example:

```
most_frequent_category = data['job'].mode()
data['job'] = data['job'].replace('unknown', most_frequent_category)
```

2. Replacing using RandomForestClassifier:

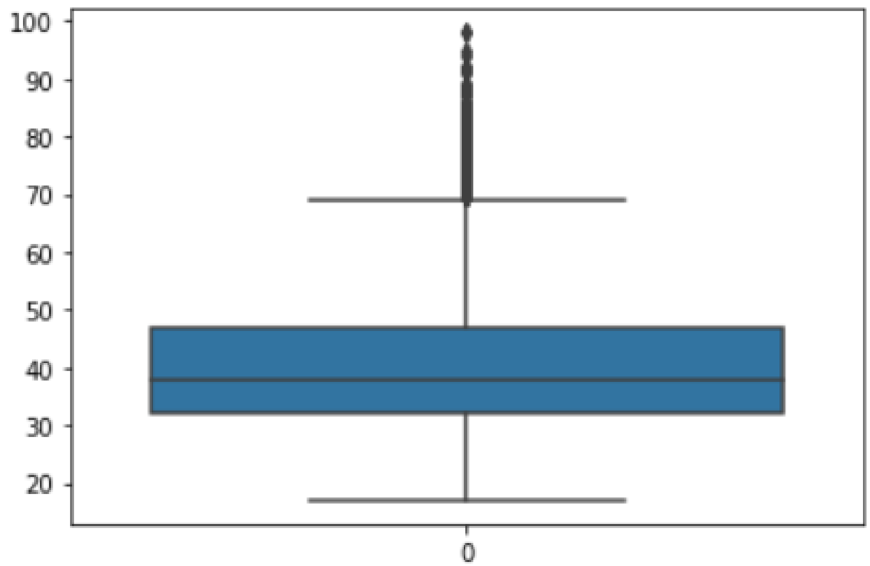
   Example:

```
# Separate known and unknown data
df_known = df[df['loan'] != 'unknown']
df_unknown = df[df['loan'] == 'unknown']

# Define categorical and numerical features
cat_features = ['job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome']
num_features = ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m

# One-hot encoding for categorical features
encoder = OneHotEncoder(handle_unknown='ignore')
encoder.fit(df_known[cat_features])
X_train_known_cat = encoder.transform(df_known[cat_features])
X_unknown_cat = encoder.transform(df_unknown[cat_features])

# Combine encoded categorical features with numerical features
X_train_known = sp.hstack((X_train_known_cat, df_known[num_features].values))
X_unknown = sp.hstack((X_unknown_cat, df_unknown[num_features].values))

# Target variable (categorical)
y_train = df_known['housing']

# Train a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_known, y_train)

# Predict the unknown values
y_unknown = clf.predict(X_unknown)

# Replace the "unknown" values in the original DataFrame with predictions
df.loc[df['housing'] == 'unknown', 'housing'] = y_unknown
```

3. Detecting and Removing Outliers:

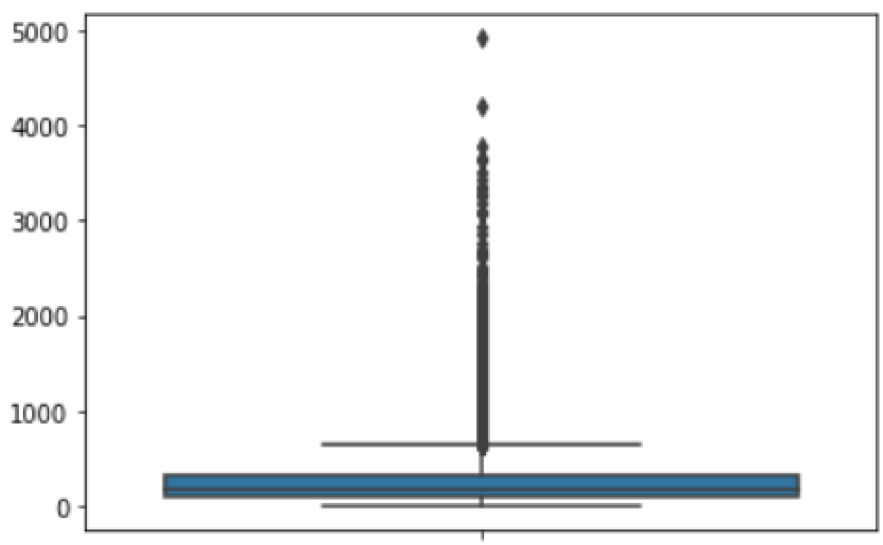
```
sns.boxplot(data.age)
```

<Axes: >




```
sns.boxplot(data.duration)
```

<Axes: >

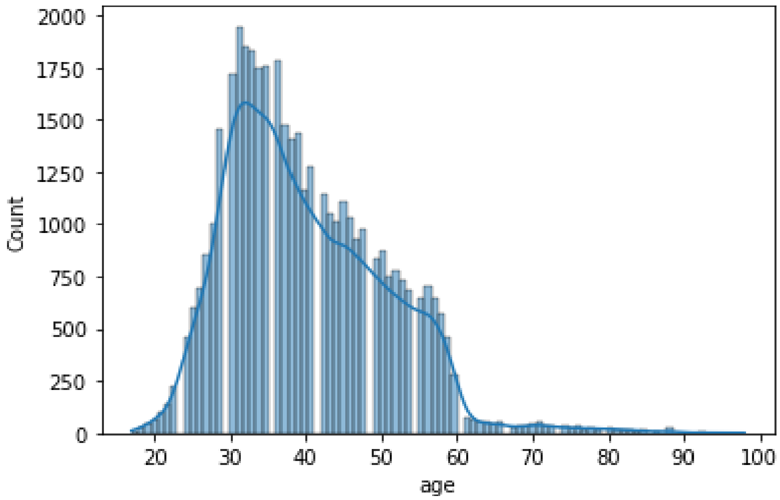


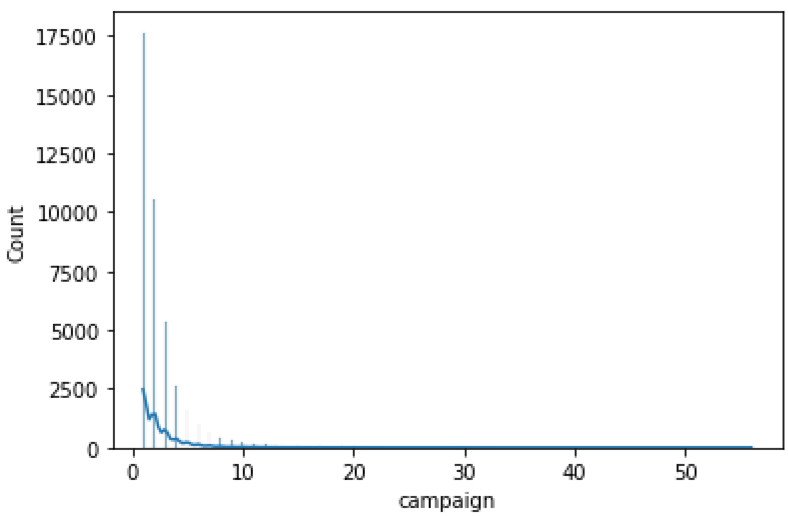
```python
def remove_outliers_iqr(df):
    outliers = {}
    for col in df.columns:
        if np.issubdtype(df[col].dtype, np.number):  # Check if the column contains numerical data
            # Calculate the IQR (Interquartile Range) for the column
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1

            # Define lower and upper bounds for outliers
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Find the indices of outliers
            outlier_indices = (df[col] < lower_bound) | (df[col] > upper_bound)

            # Create a DataFrame containing the outliers
            col_outliers = df[outlier_indices]

            # Add the outliers to the dictionary
            outliers[col] = col_outliers

    # Remove the rows containing outliers from the original DataFrame
    for col, col_outliers in outliers.items():
        df = df[~df.index.isin(col_outliers.index)]

    # Print information about removed outliers and the new shape
    if outliers:
        print('These outliers have been removed from your dataset:')
        for col, col_outliers in outliers.items():
            print(f'\nOutliers in column "{col}":')
            print(col_outliers)
    else:
        print('No outliers were found in the dataset.')

    print('\nNew shape is:', df.shape)
    return df
```

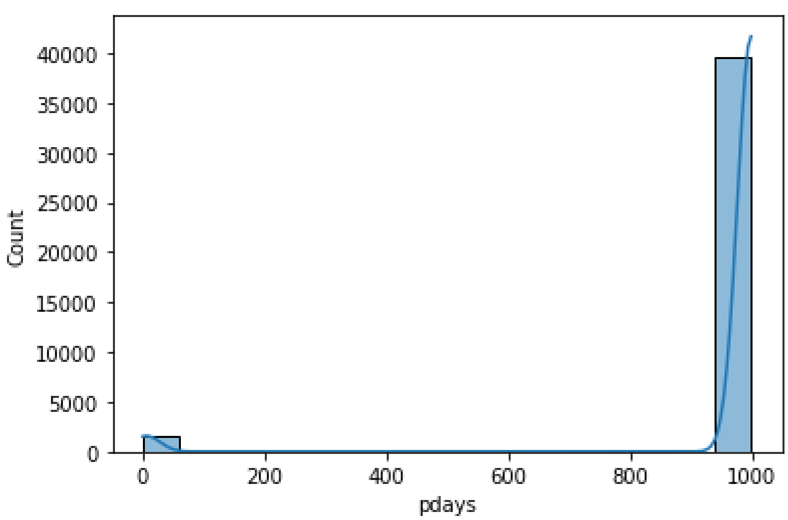
Some EDA on the data:

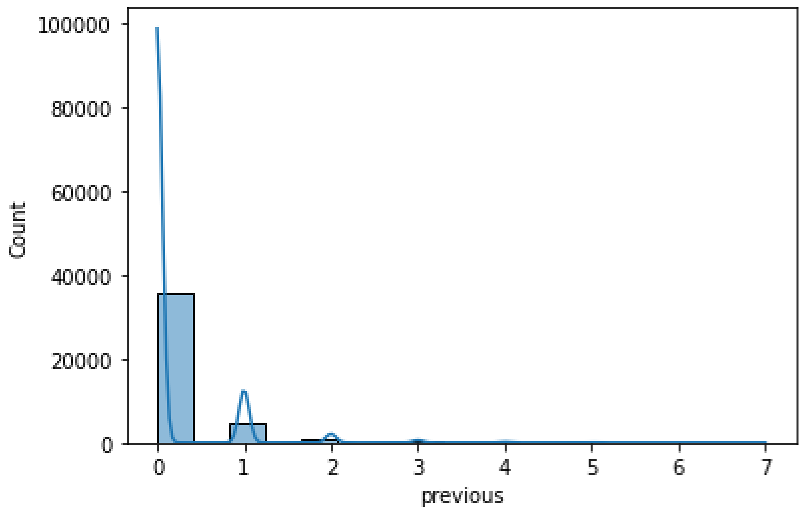
I tried several EDA approaches, which will be manifested as follows. Improving the understanding of features, this analysis looks at both categorical and continuous data. By studying the connections and patterns in the data, to get a clearer view. This explanation shows why I used feature analysis in EDA and focuses on the results and what have been learned from them. Among clients with different ages, education backgrounds, marital statuses, etc, I plan to seek out how many of them subscribe to the terms under each category. To examine the distribution of categorical data, I have used pie chart whereas in case of continuous data I have used histogram to discover the patterns within the data

Distribution of Numerical Data

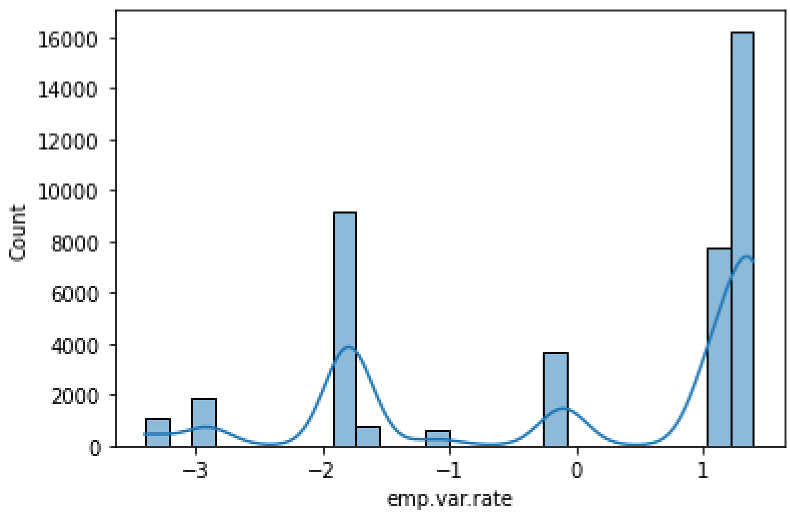




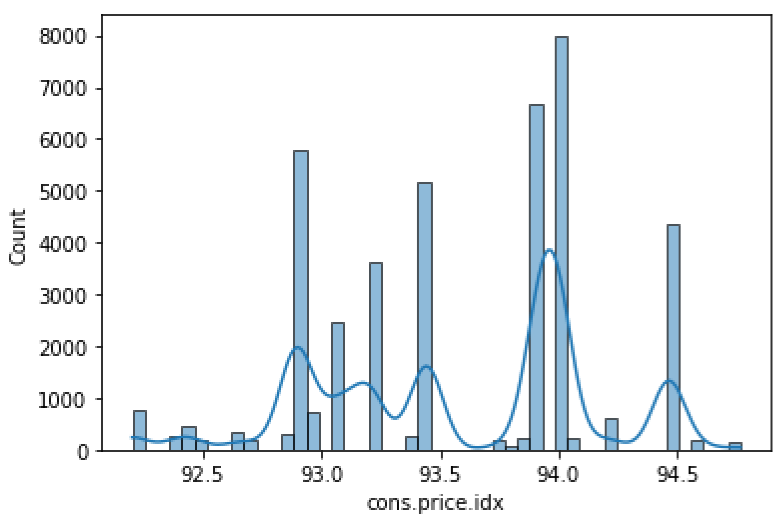




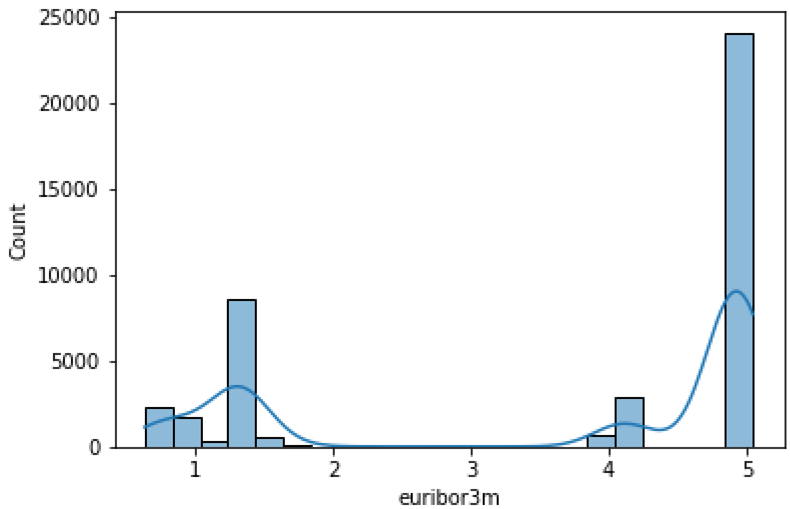
100000
80000
60000
40000
20000
0
Count
0
1
2
3
4
5
6
7
previous

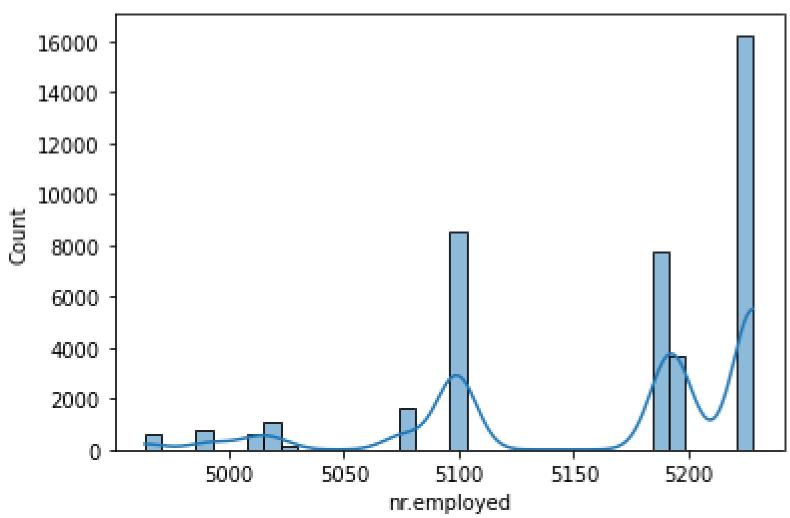


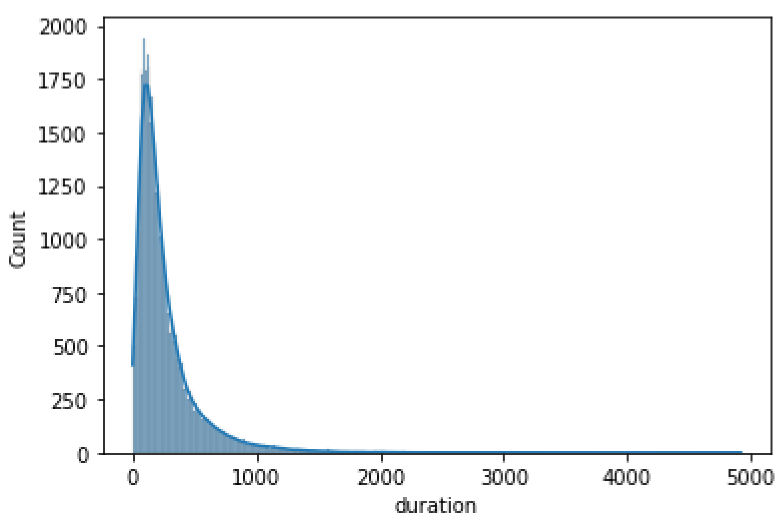
16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−3
−2
−1
0
1
emp.var.rate



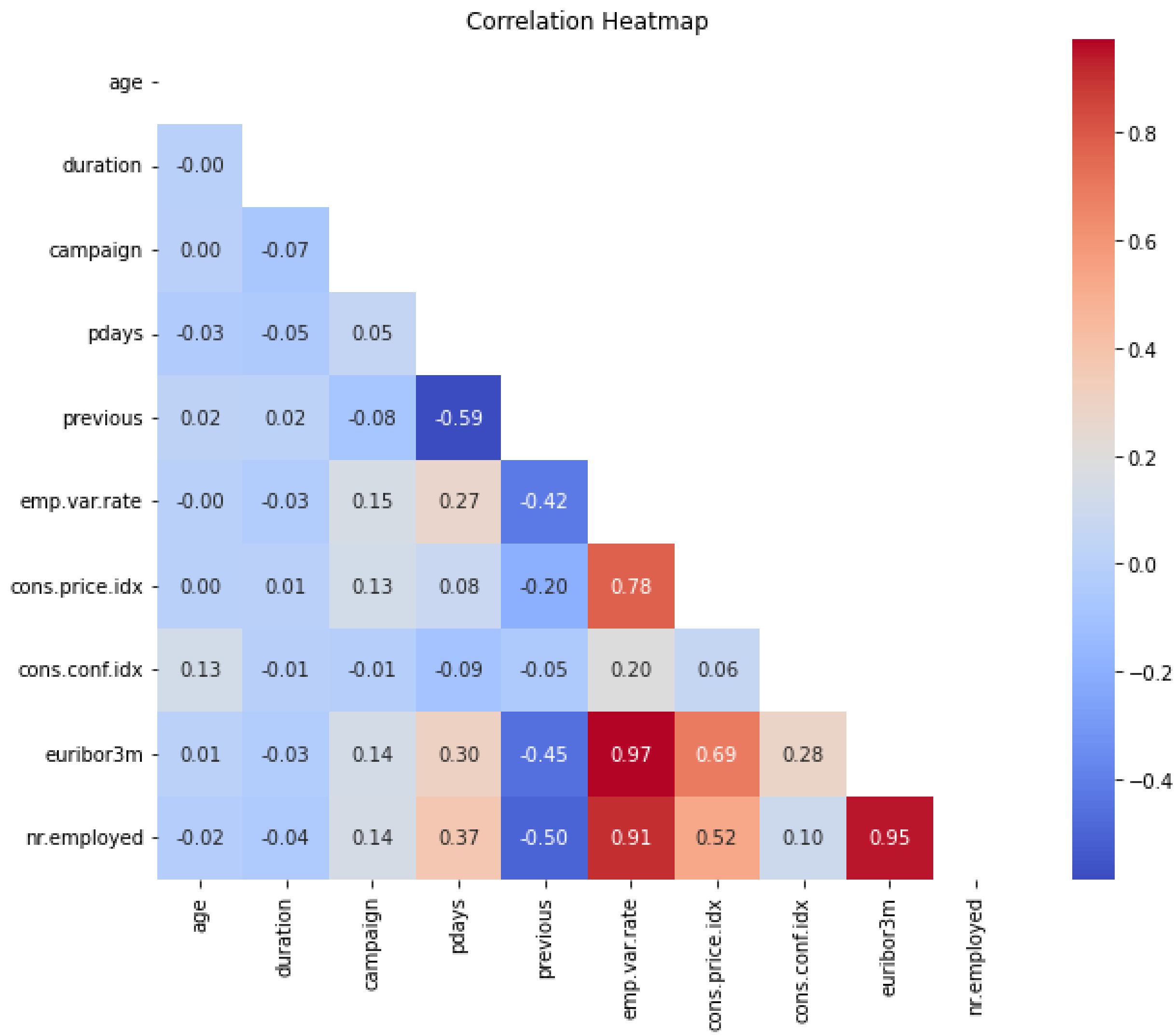
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
92.5
93.0
93.5
94.0
94.5
cons.price.idx


25000
20000
15000
10000
5000
0
Count
1
2
3
4
5
euribor3m

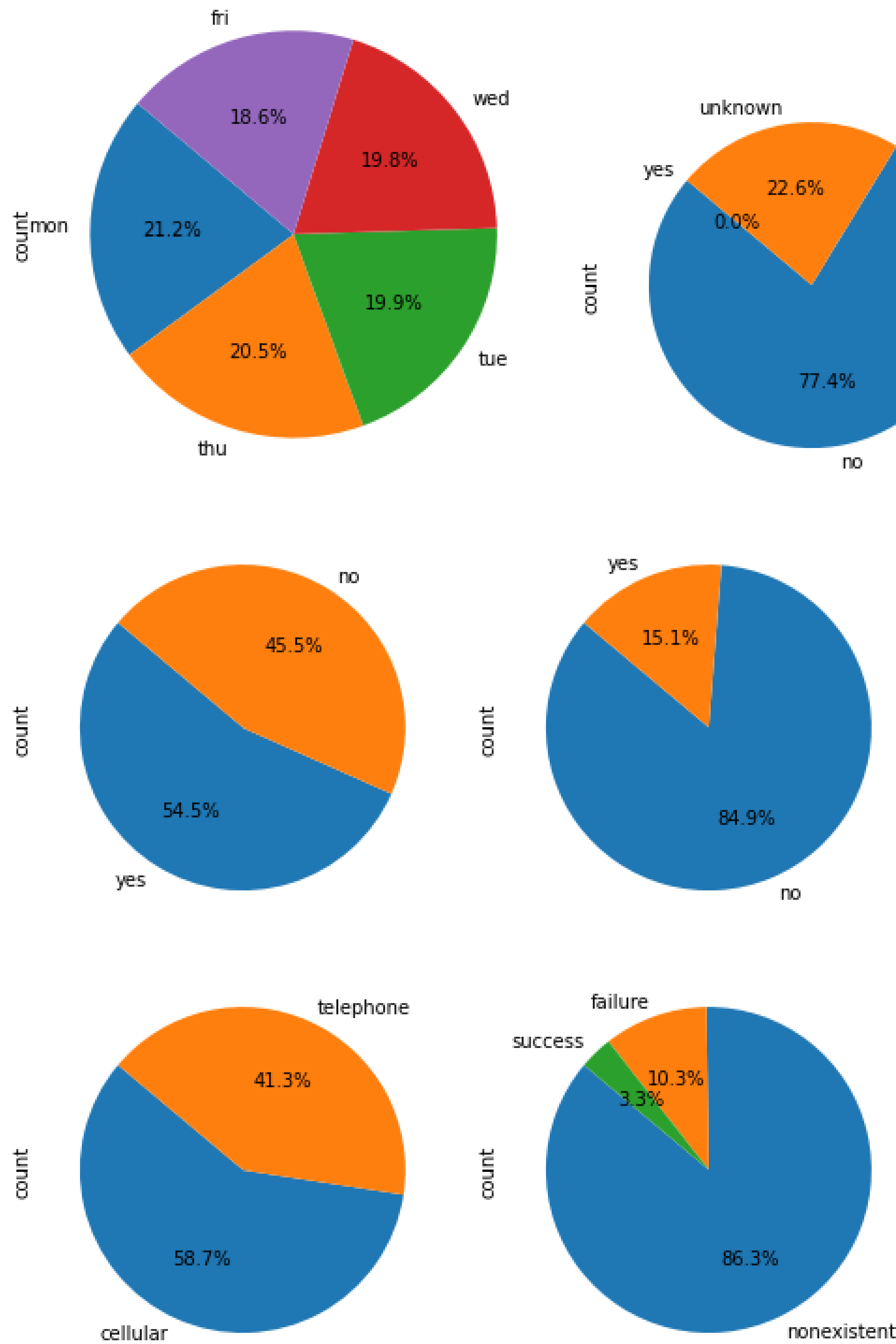

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
5000
5050
5100
5150
5200
nr.employed

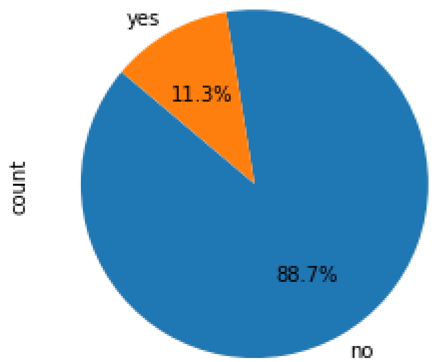

2000
1750
1500
1250
1000
750
500
250
0
Count
0
1000
2000
3000
4000
5000
duration

Correlation Heatmap:

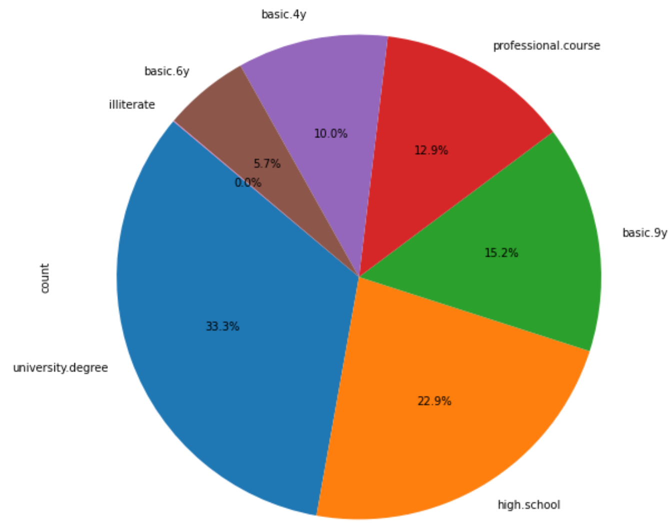


There is a high correlation between emp.var.rate and nr.employed, equibor3m and nr.employed, Distribution of Categorical Data:

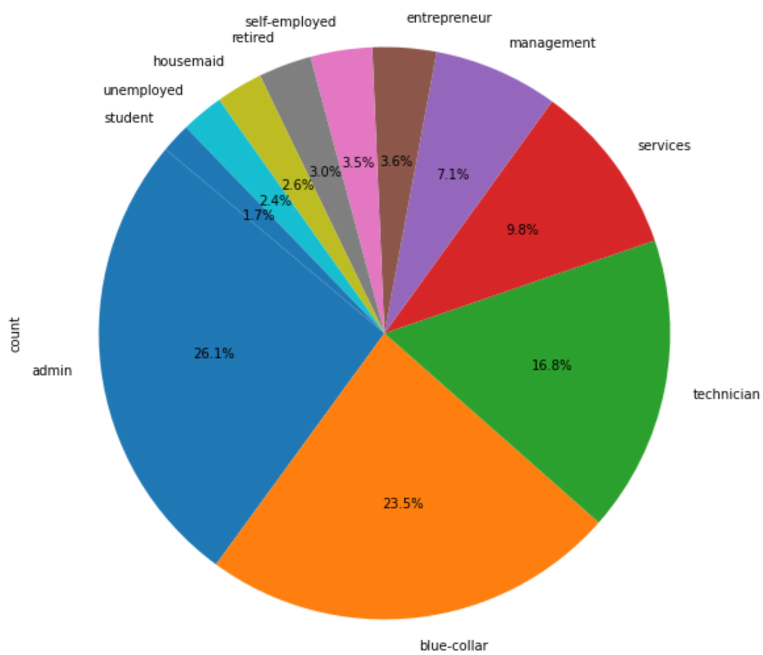




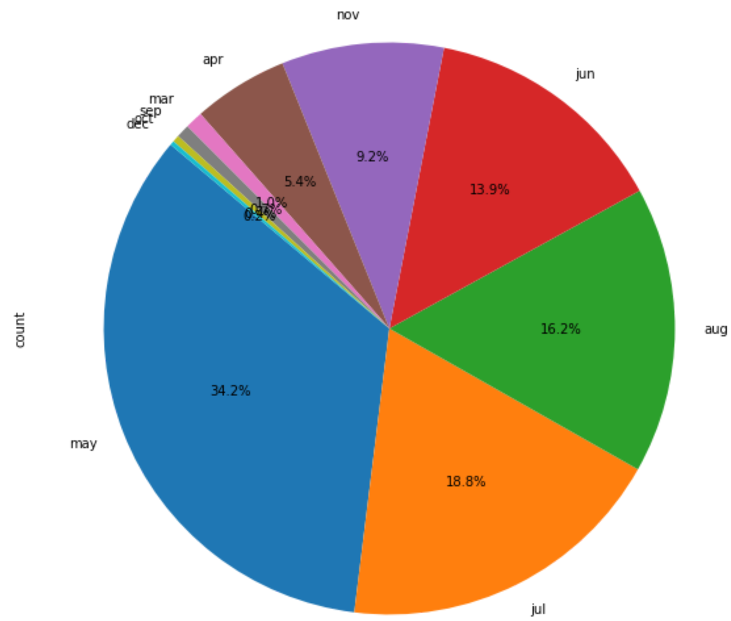
count
yes
11.3%
88.7%
no



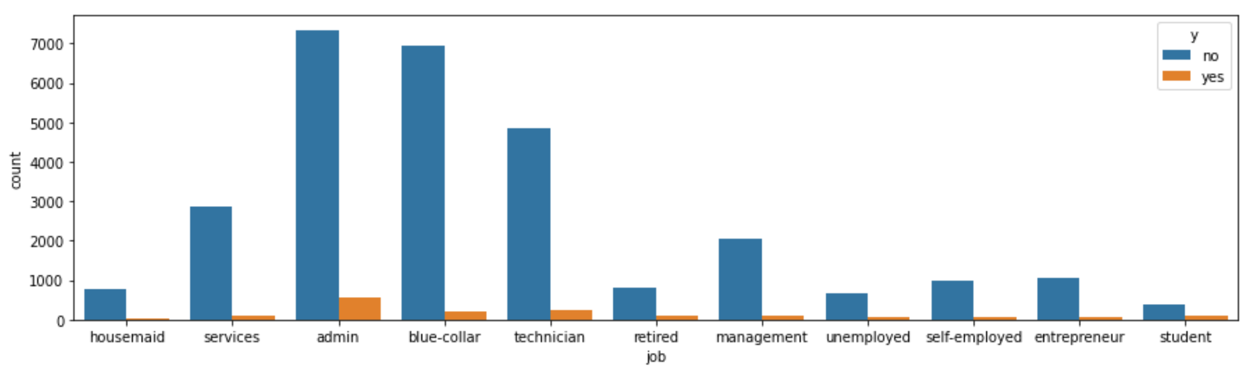
count
basic.4y
professional.course
basic.6y
illiterate
10.0%
12.9%
5.7%
0.0%
basic.9y
15.2%
33.3%
university.degree
22.9%
high.school

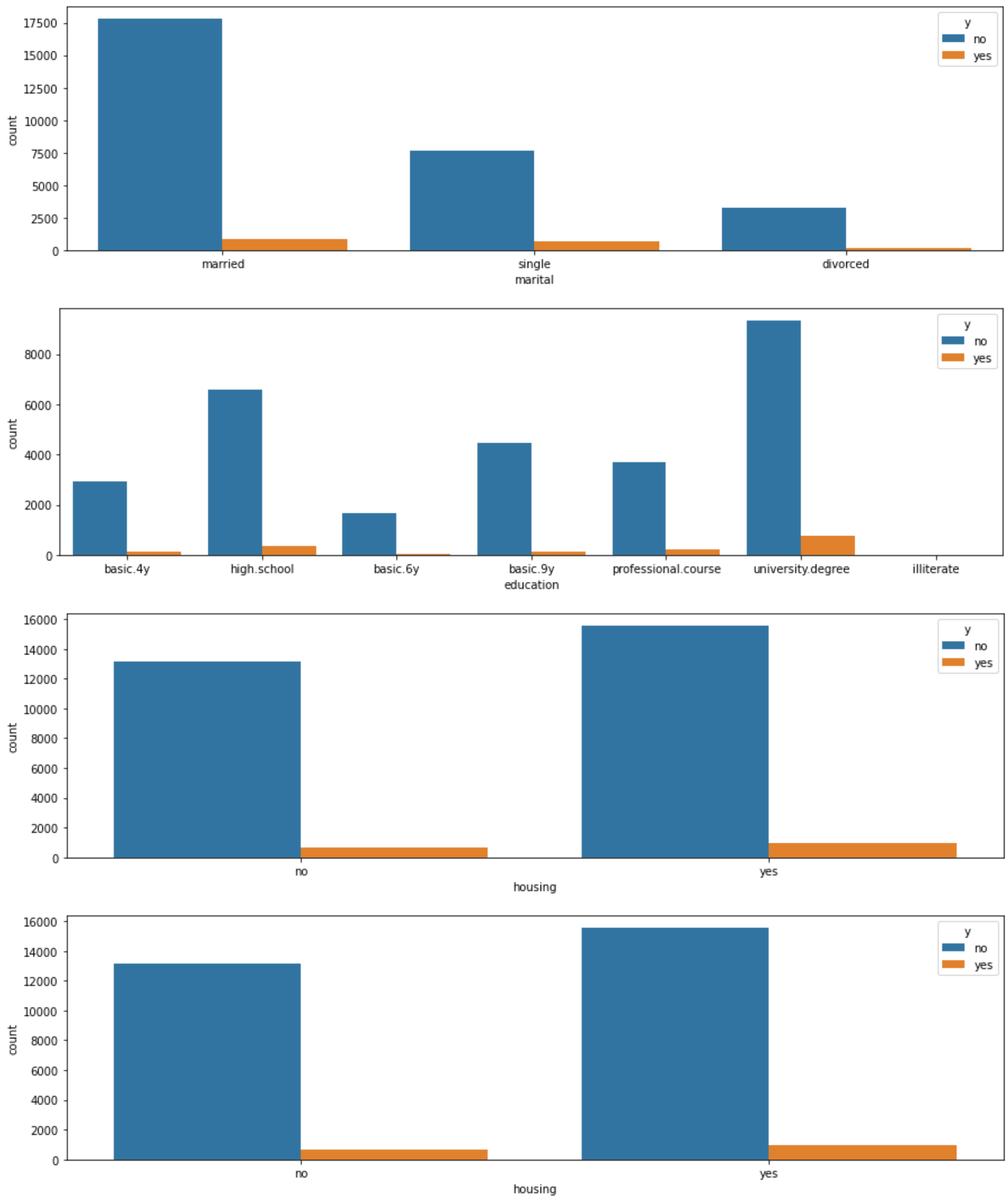


count
self-employed
entrepreneur
retired
management
housemaid
unemployed
student
services
3.5%
3.6%
3.0%
7.1%
2.6%
2.4%
1.7%
9.8%
26.1%
admin
16.8%
technician
23.5%
blue-collar



Relationship of categorical data with output result





I plotted 4 bar charts containing the number of people subscribing to the terms with different ages, jobs, educations, loan, marital statuses, and I found out for each of these attributes, the number of people who do not subscribe to the terms surpasses the number of people who subscribe to it.

Models Tried

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.944664	0.545455	0.206897	0.300000
1	Random Forest Classifier	0.947299	0.587500	0.270115	0.370079
2	Gradient Boosting Classifier	0.949440	0.613260	0.318966	0.419660
3	AdaBoost Classifier	0.944664	0.534091	0.270115	0.358779
4	Bagging Classifier	0.943841	0.515419	0.336207	0.406957
5	Extra Trees Classifier	0.944170	0.524590	0.275862	0.361582
6	Support Vector Classifier	0.942688	0.000000	0.000000	0.000000
7	K-Nearest Neighbors (KNN)	0.943511	0.513661	0.270115	0.354049
8	Naive Bayes (GaussianNB)	0.918972	0.364662	0.557471	0.440909
9	Decision Tree Classifier	0.932148	0.412088	0.431034	0.421348

Due to imbalance in data we can clearly see that precision and recall is less for all models tried.

So, I decided to upsample the less present data of no cases.

After Upsampling the data

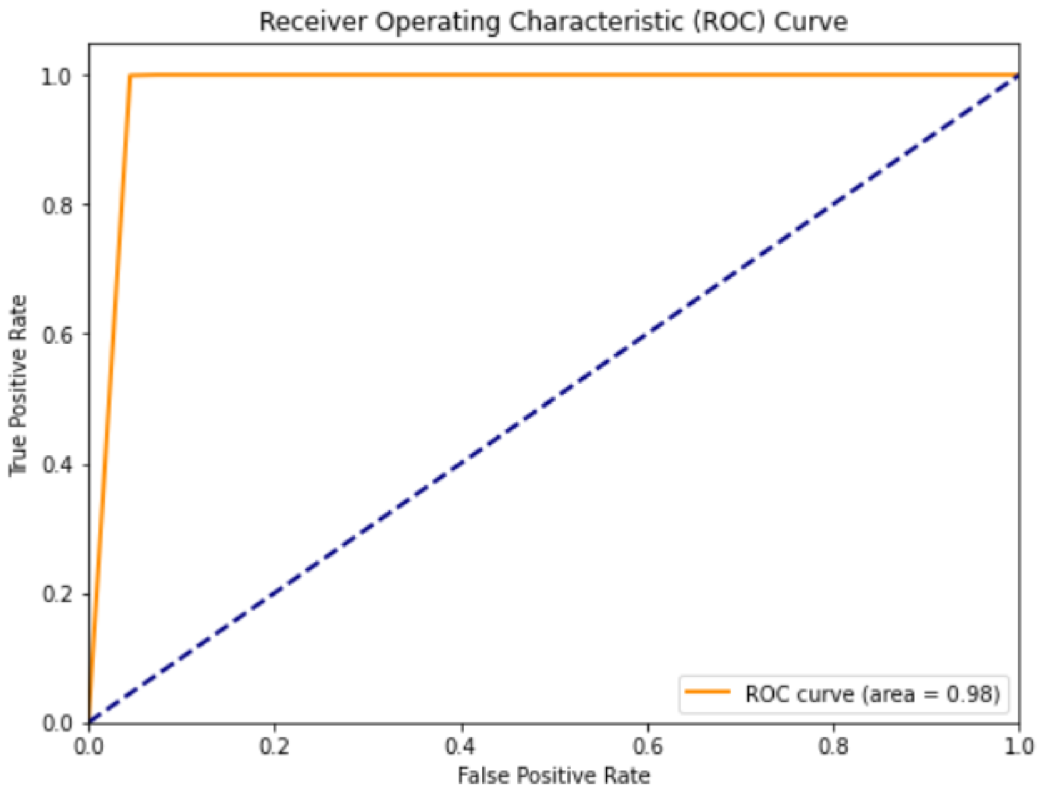
	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.880160	0.863496	0.903891	0.883232
1	Random Forest Classifier	0.985020	0.970992	1.000000	0.985282
2	Gradient Boosting Classifier	0.892876	0.864363	0.932723	0.897243
3	AdaBoost Classifier	0.877315	0.878496	0.876563	0.877529
4	Bagging Classifier	0.982756	0.966752	1.000000	0.983095
5	Extra Trees Classifier	0.989665	0.979805	1.000000	0.989799
6	Support Vector Classifier	0.928526	0.892089	0.975452	0.931910
7	K-Nearest Neighbors (KNN)	0.949486	0.909254	0.998958	0.951997
8	Naive Bayes (GaussianNB)	0.799803	0.800928	0.799444	0.800185
9	Decision Tree Classifier	0.978227	0.958384	1.000000	0.978750

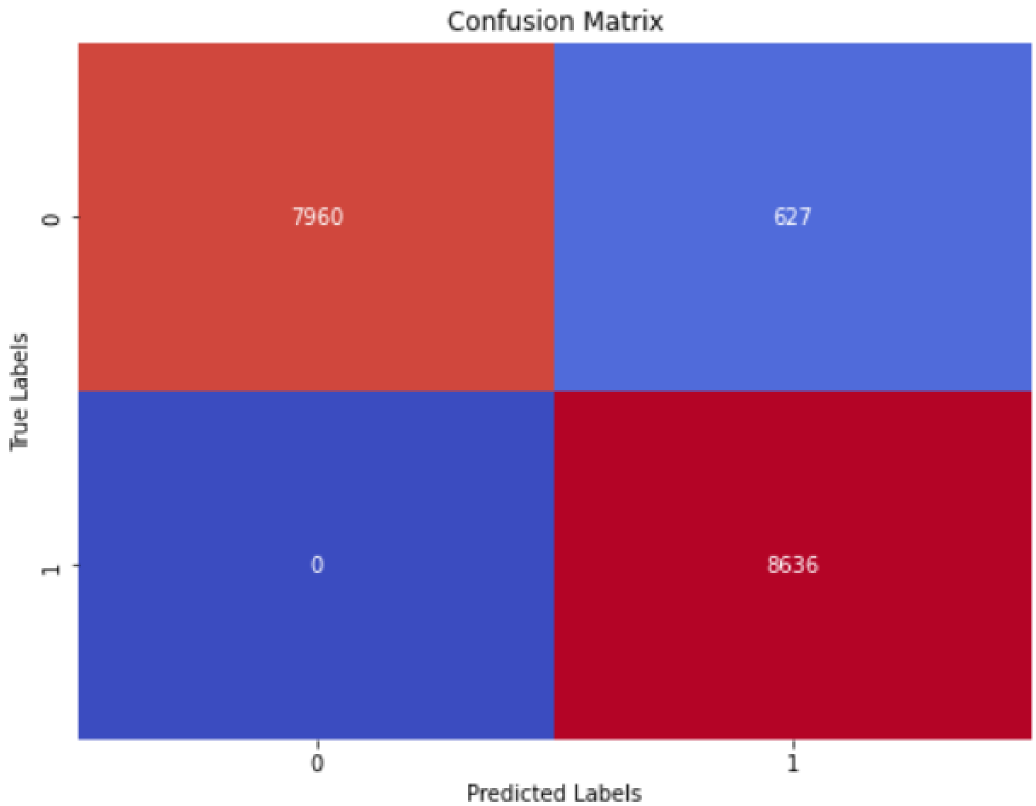
After hyper-parameter tuning KNN:

```
Best Parameters for KNN: {'model  weights': 'uniform', 'model  p': 1, 'model  n neighbors': 3, 'model  algorithm': 'auto'}
```

	Model	Accuracy	Precision	Recall	F1 Score
0	K-Nearest Neighbors (KNN)	0.962957	0.931206	1.0	0.964377

Other Results:





\K fold cross validation accuracy:

```
Accuracy Scores for Each Fold:
Fold 1: 0.9651
Fold 2: 0.7965
Fold 3: 0.9448
Fold 4: 0.7500
Fold 5: 0.5369
Mean Accuracy: 0.799
```

Model Implementation: