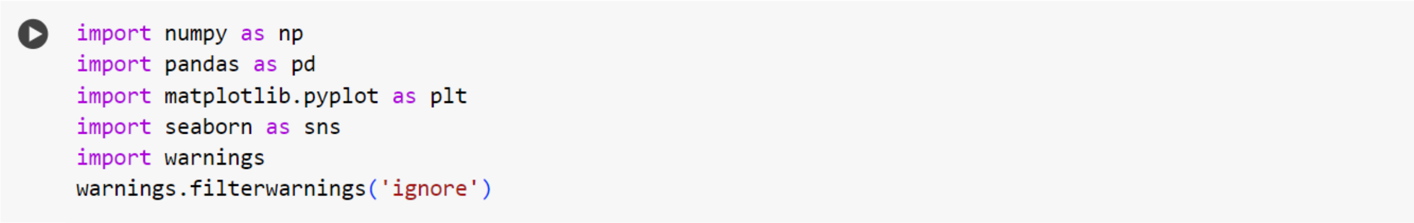
Prodigy InfoTech Internship

Task 3

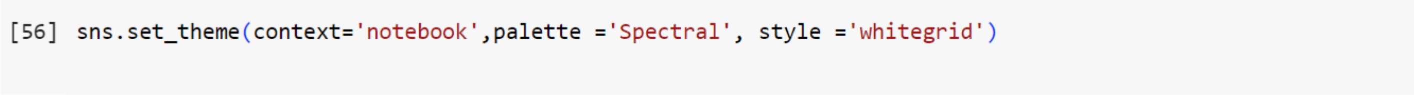
Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioural data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository.

Sample Dataset :- Bank Marketing

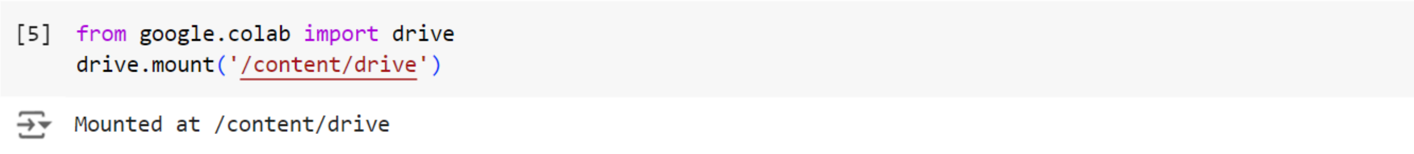
Loading dataset:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

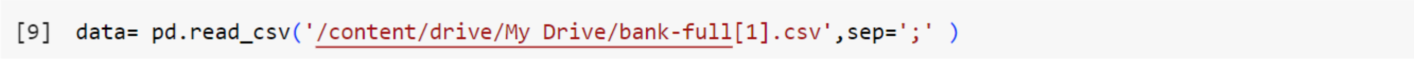


```
[56] sns.set_theme(context='notebook',palette ='Spectral', style ='whitegrid')
```



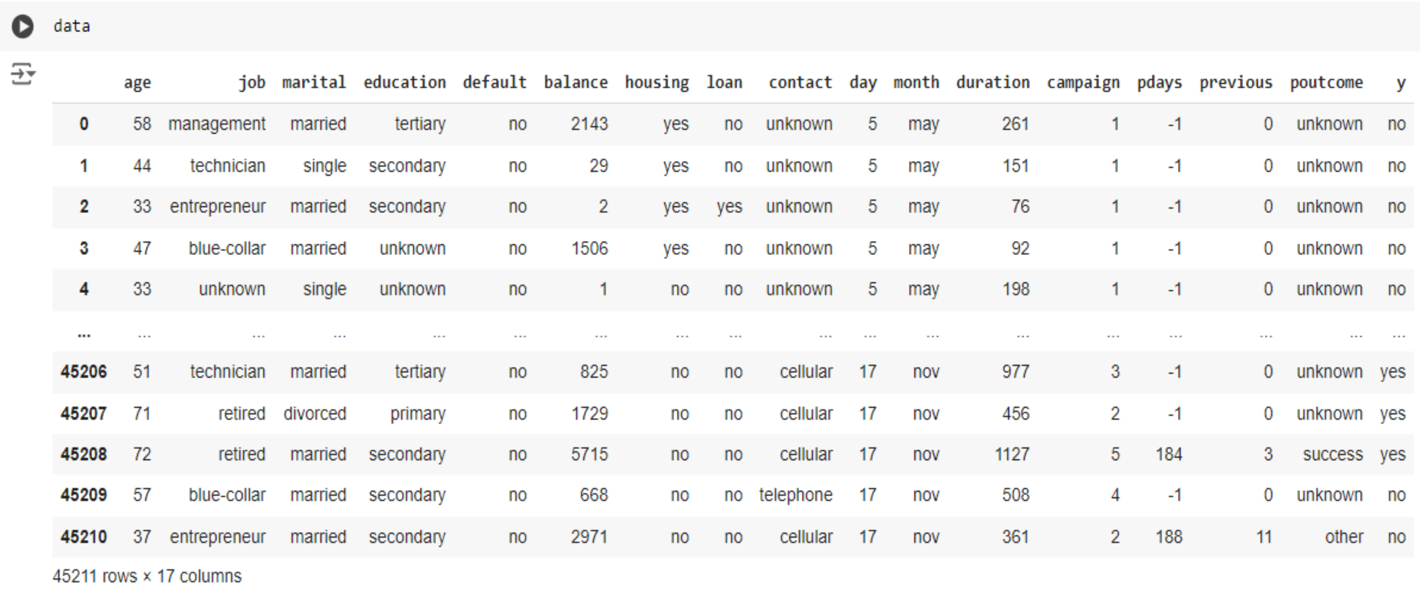
```
[5]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```



```
[9]  data= pd.read_csv('/content/drive/My Drive/bank-full[1].csv',sep=';' )
```

Understanding the shape of the data:

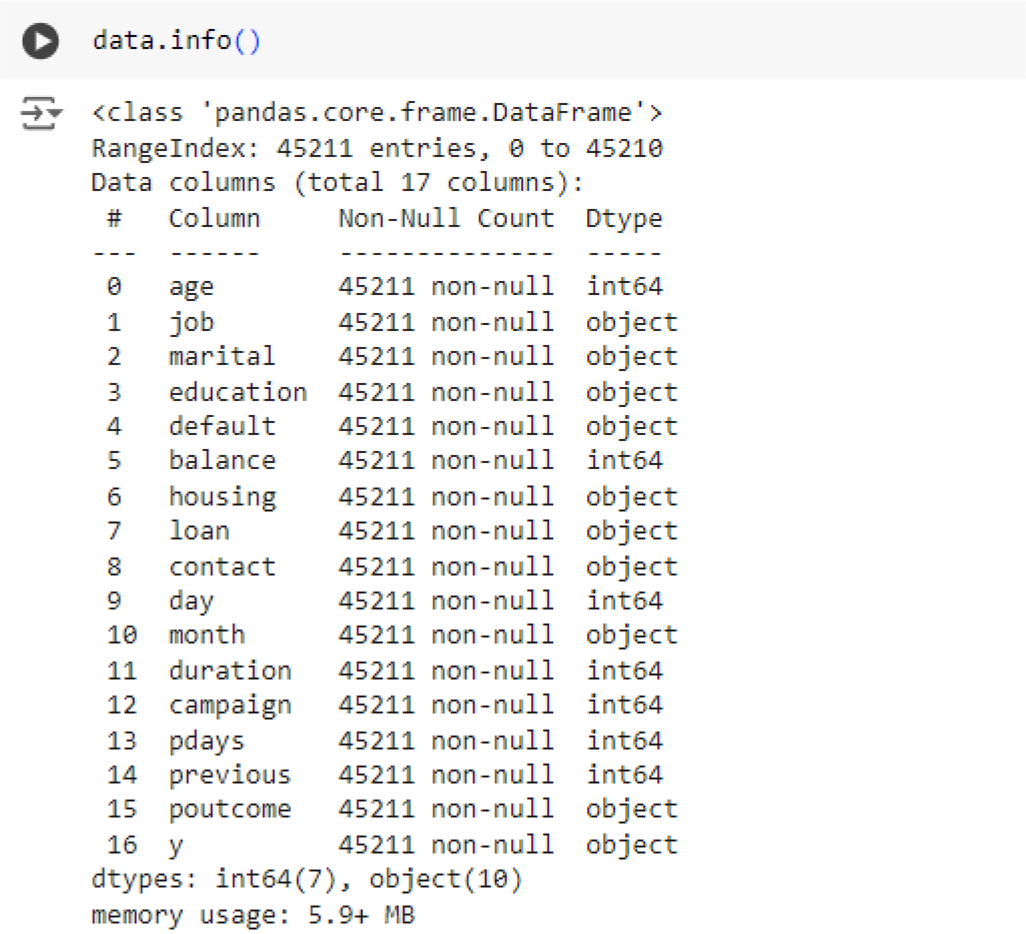


```
data
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

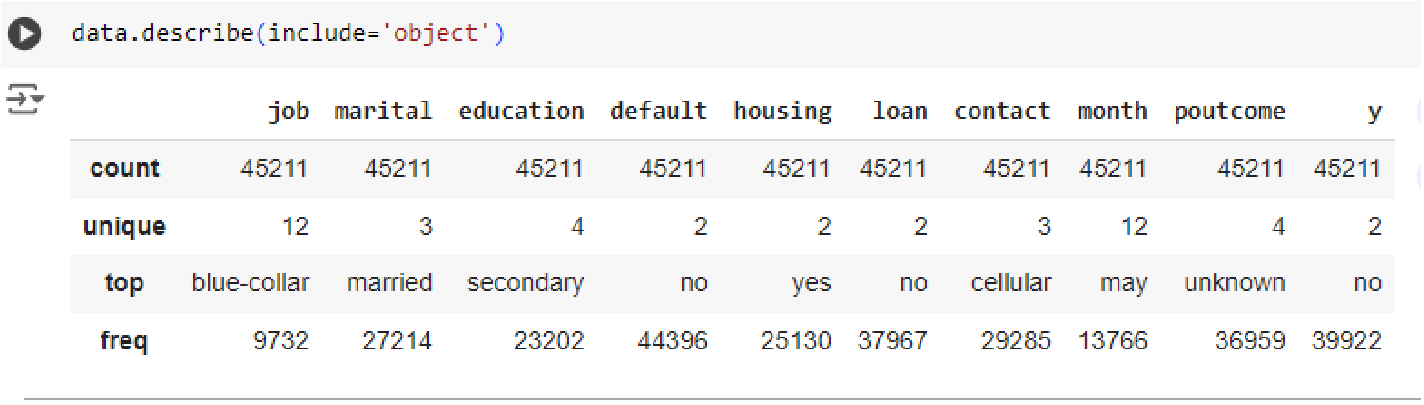
45211 rows × 17 columns

Aditya Rathore
linkedIn



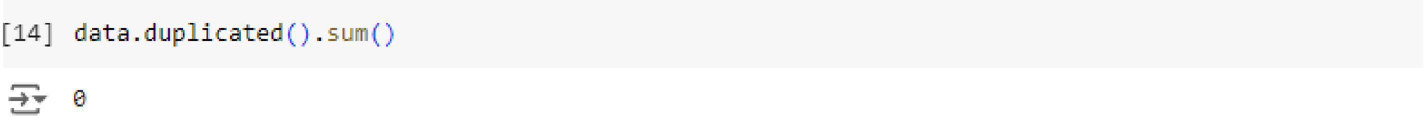
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```



```
data.describe(include='object')
```

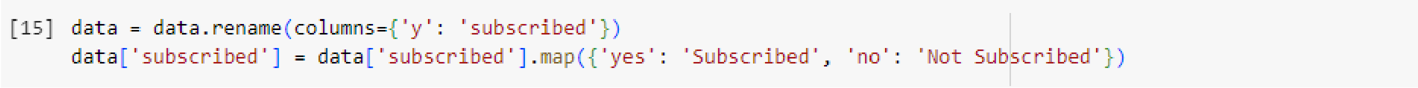
	job	marital	education	default	housing	loan	contact	month	poutcome	y
count	45211	45211	45211	45211	45211	45211	45211	45211	45211	45211
unique	12	3	4	2	2	2	3	12	4	2
top	blue-collar	married	secondary	no	yes	no	cellular	may	unknown	no
freq	9732	27214	23202	44396	25130	37967	29285	13766	36959	39922



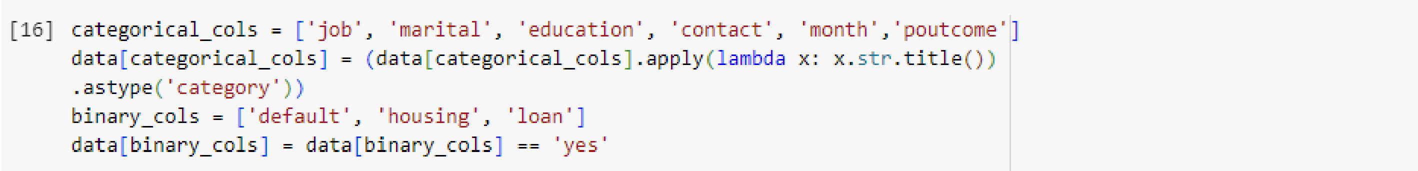
```
[14] data.duplicated().sum()
```

```
0
```

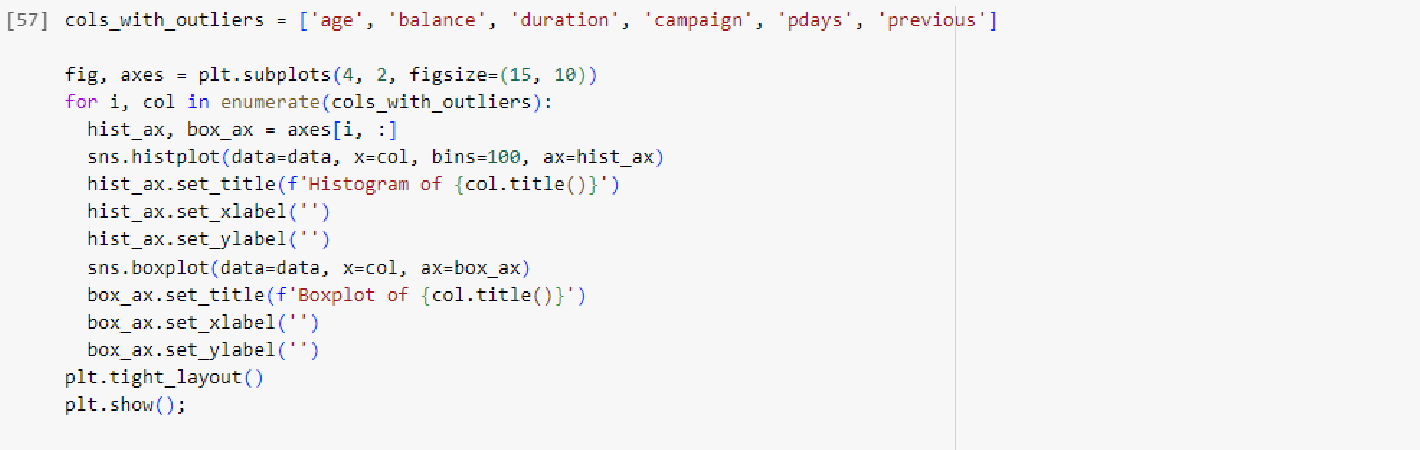
Data Cleaning:



```
[15]  data = data.rename(columns={'y': 'subscribed'})
      data['subscribed'] = data['subscribed'].map({'yes': 'Subscribed', 'no': 'Not Subscribed'})
```

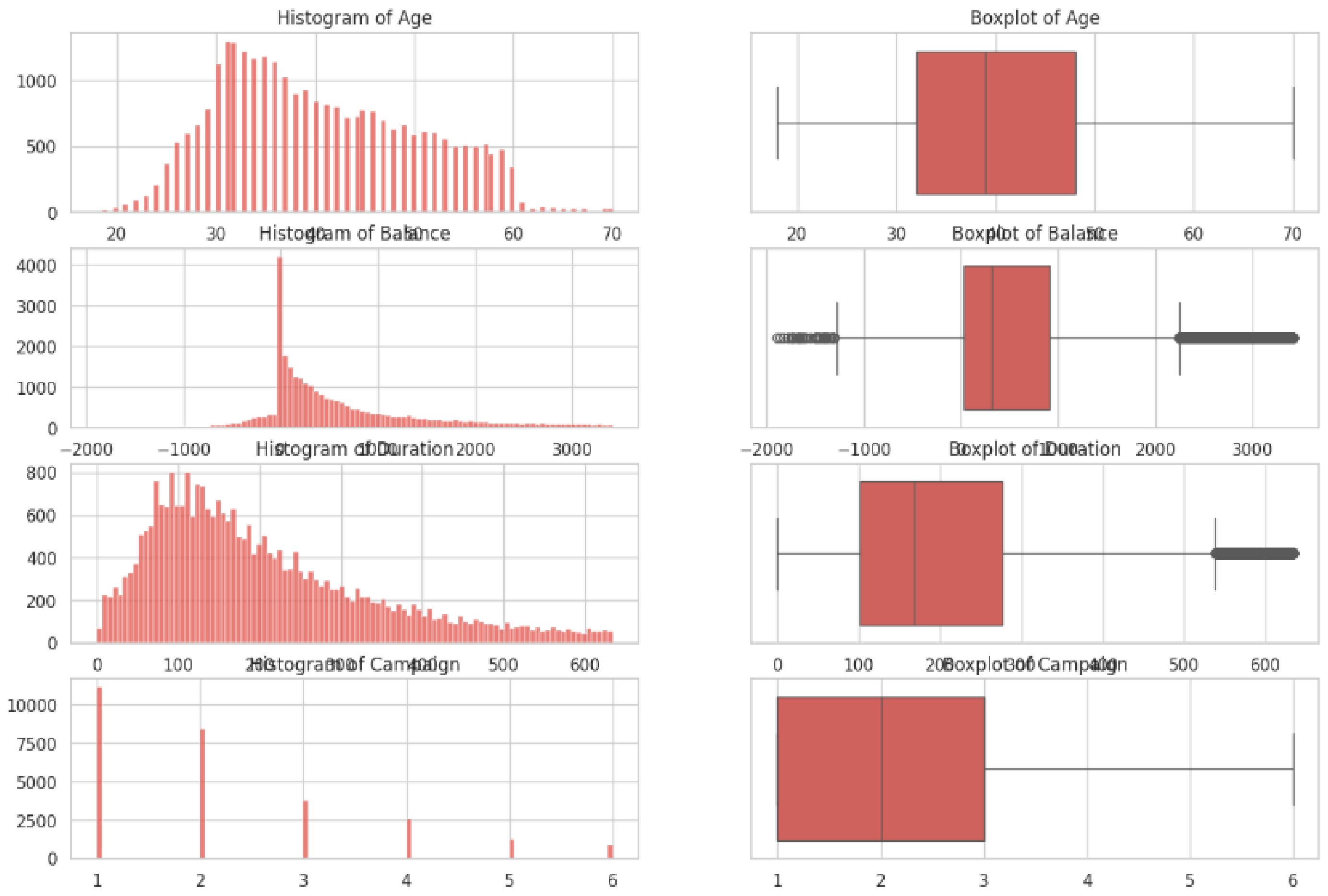


```
[16]  categorical_cols = ['job', 'marital', 'education', 'contact', 'month','poutcome']
      data[categorical_cols] = (data[categorical_cols].apply(lambda x: x.str.title())
      .astype('category'))
      binary_cols = ['default', 'housing', 'loan']
      data[binary_cols] = data[binary_cols] == 'yes'
```



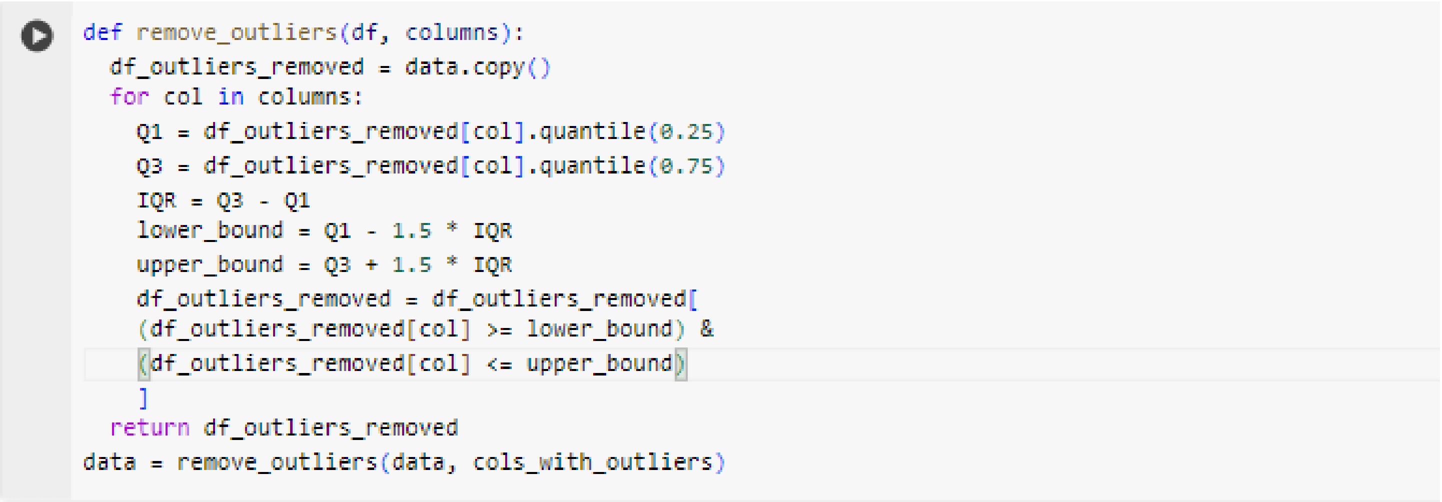
```
[57] cols_with_outliers = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']

     fig, axes = plt.subplots(4, 2, figsize=(15, 10))
     for i, col in enumerate(cols_with_outliers):
       hist_ax, box_ax = axes[i, :]
       sns.histplot(data=data, x=col, bins=100, ax=hist_ax)
       hist_ax.set_title(f'Histogram of {col.title()}')
       hist_ax.set_xlabel('')
       hist_ax.set_ylabel('')
       sns.boxplot(data=data, x=col, ax=box_ax)
       box_ax.set_title(f'Boxplot of {col.title()}')
       box_ax.set_xlabel('')
       box_ax.set_ylabel('')
     plt.tight_layout()
     plt.show();
```

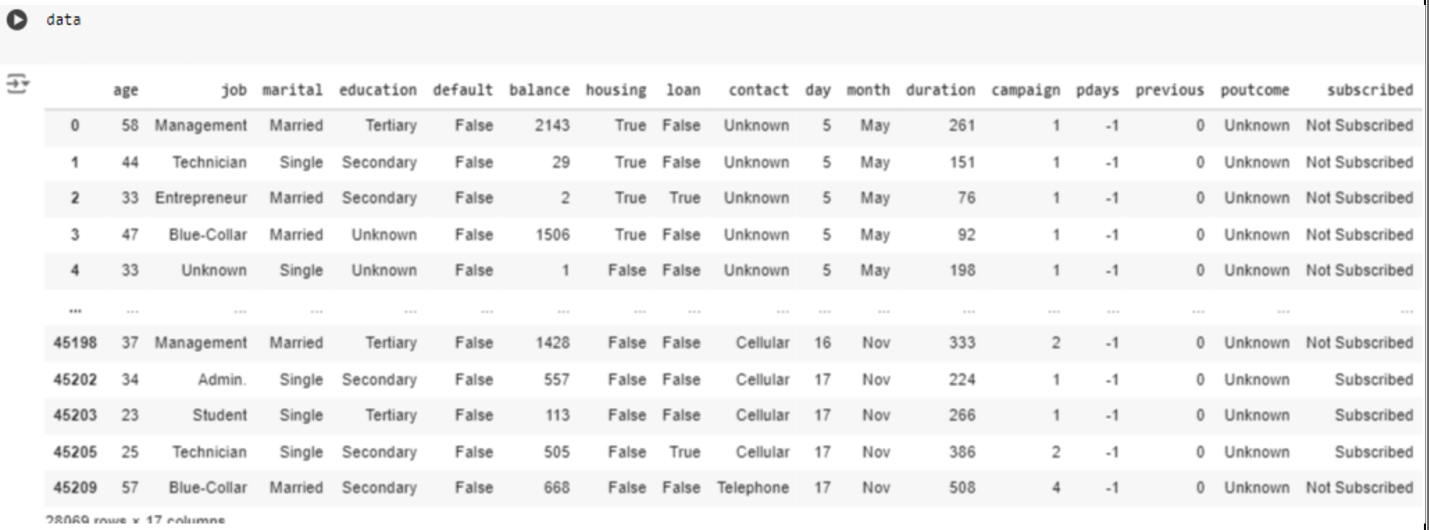


Aditya Rathore

linkedIn



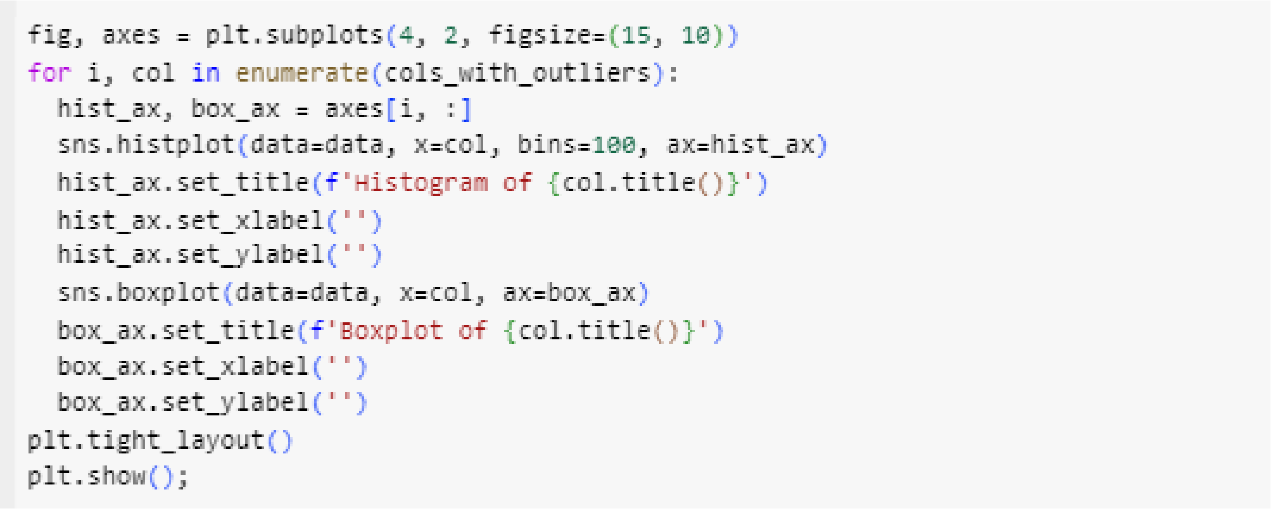
```python
def remove_outliers(df, columns):
  df_outliers_removed = data.copy()
  for col in columns:
    Q1 = df_outliers_removed[col].quantile(0.25)
    Q3 = df_outliers_removed[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_outliers_removed = df_outliers_removed[
    (df_outliers_removed[col] >= lower_bound) &
    (df_outliers_removed[col] <= upper_bound)
    ]
  return df_outliers_removed
data = remove_outliers(data, cols_with_outliers)
```



```python
data
```

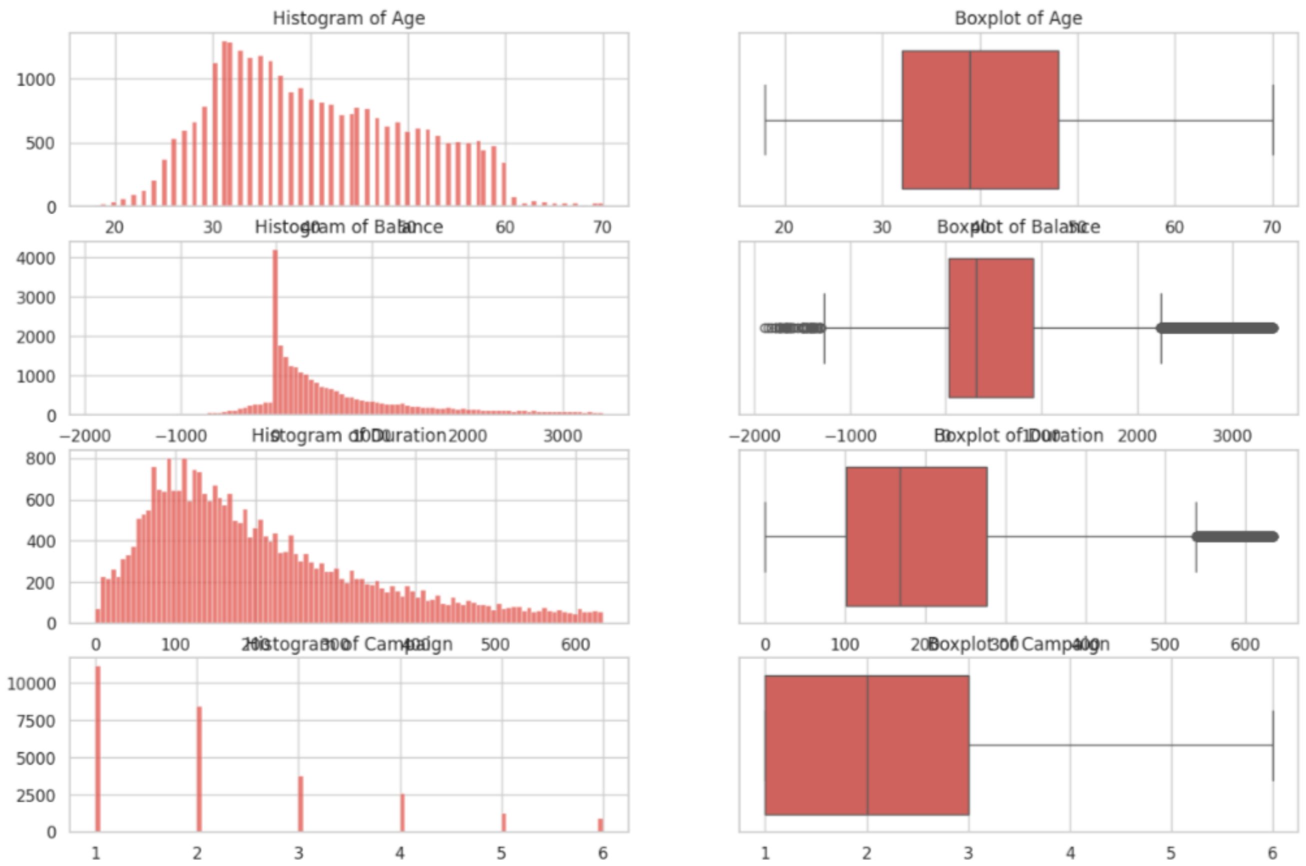
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	subscribed
0	58	Management	Married	Tertiary	False	2143	True	False	Unknown	5	May	261	1	-1	0	Unknown	Not Subscribed
1	44	Technician	Single	Secondary	False	29	True	False	Unknown	5	May	151	1	-1	0	Unknown	Not Subscribed
2	33	Entrepreneur	Married	Secondary	False	2	True	True	Unknown	5	May	76	1	-1	0	Unknown	Not Subscribed
3	47	Blue-Collar	Married	Unknown	False	1506	True	False	Unknown	5	May	92	1	-1	0	Unknown	Not Subscribed
4	33	Unknown	Single	Unknown	False	1	False	False	Unknown	5	May	198	1	-1	0	Unknown	Not Subscribed
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45198	37	Management	Married	Tertiary	False	1428	False	False	Cellular	16	Nov	333	2	-1	0	Unknown	Not Subscribed
45202	34	Admin.	Single	Secondary	False	557	False	False	Cellular	17	Nov	224	1	-1	0	Unknown	Subscribed
45203	23	Student	Single	Tertiary	False	113	False	False	Cellular	17	Nov	266	1	-1	0	Unknown	Subscribed
45205	25	Technician	Single	Secondary	False	505	False	True	Cellular	17	Nov	386	2	-1	0	Unknown	Subscribed
45209	57	Blue-Collar	Married	Secondary	False	668	False	False	Telephone	17	Nov	508	4	-1	0	Unknown	Not Subscribed

28069 rows x 17 columns

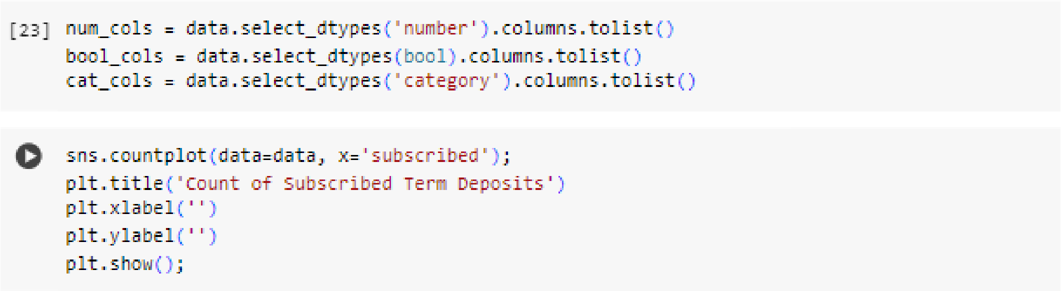


```python
fig, axes = plt.subplots(4, 2, figsize=(15, 10))
for i, col in enumerate(cols_with_outliers):
  hist_ax, box_ax = axes[i, :]
  sns.histplot(data=data, x=col, bins=100, ax=hist_ax)
  hist_ax.set_title(f'Histogram of {col.title()}')
  hist_ax.set_xlabel('')
  hist_ax.set_ylabel('')
  sns.boxplot(data=data, x=col, ax=box_ax)
  box_ax.set_title(f'Boxplot of {col.title()}')
  box_ax.set_xlabel('')
  box_ax.set_ylabel('')
plt.tight_layout()
plt.show();
```

Aditya Rathore
linkedIn

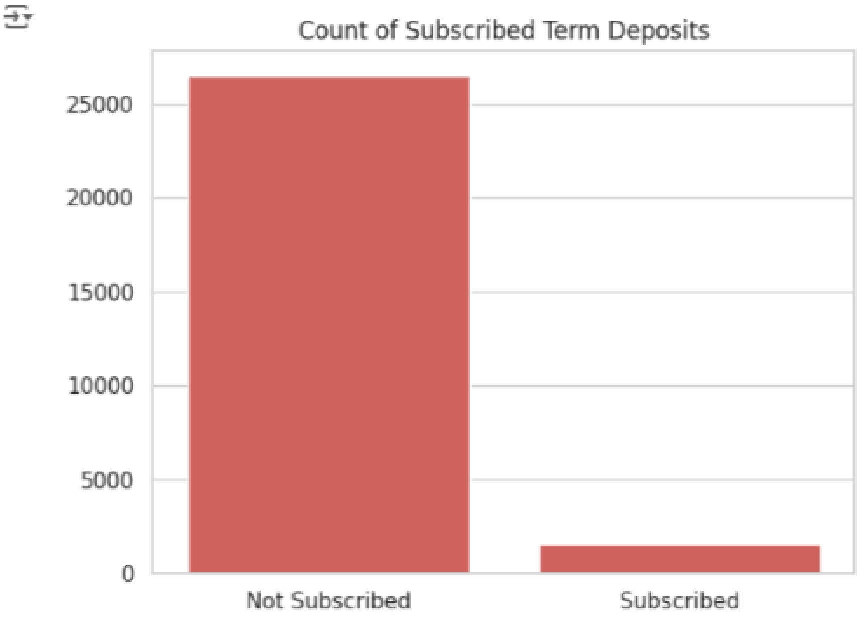


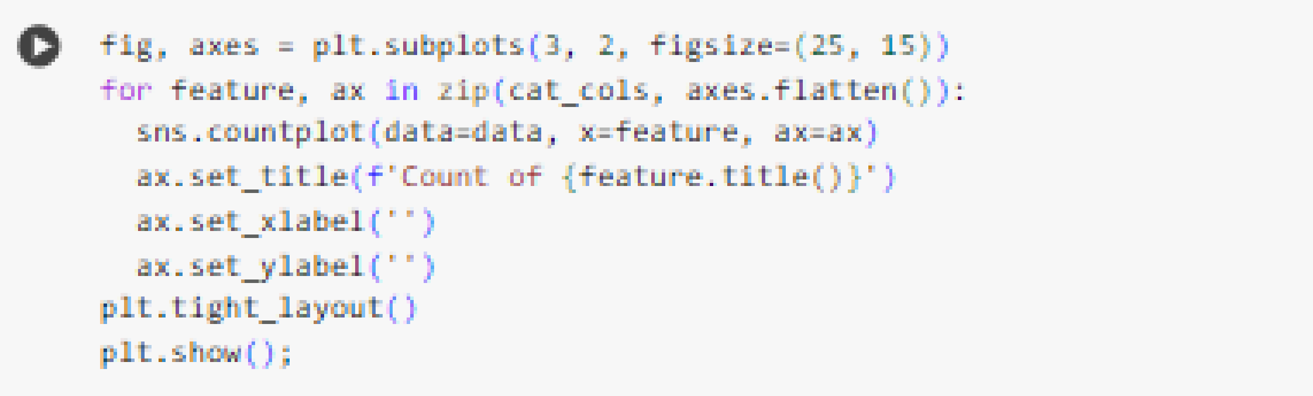
Data Exploration:



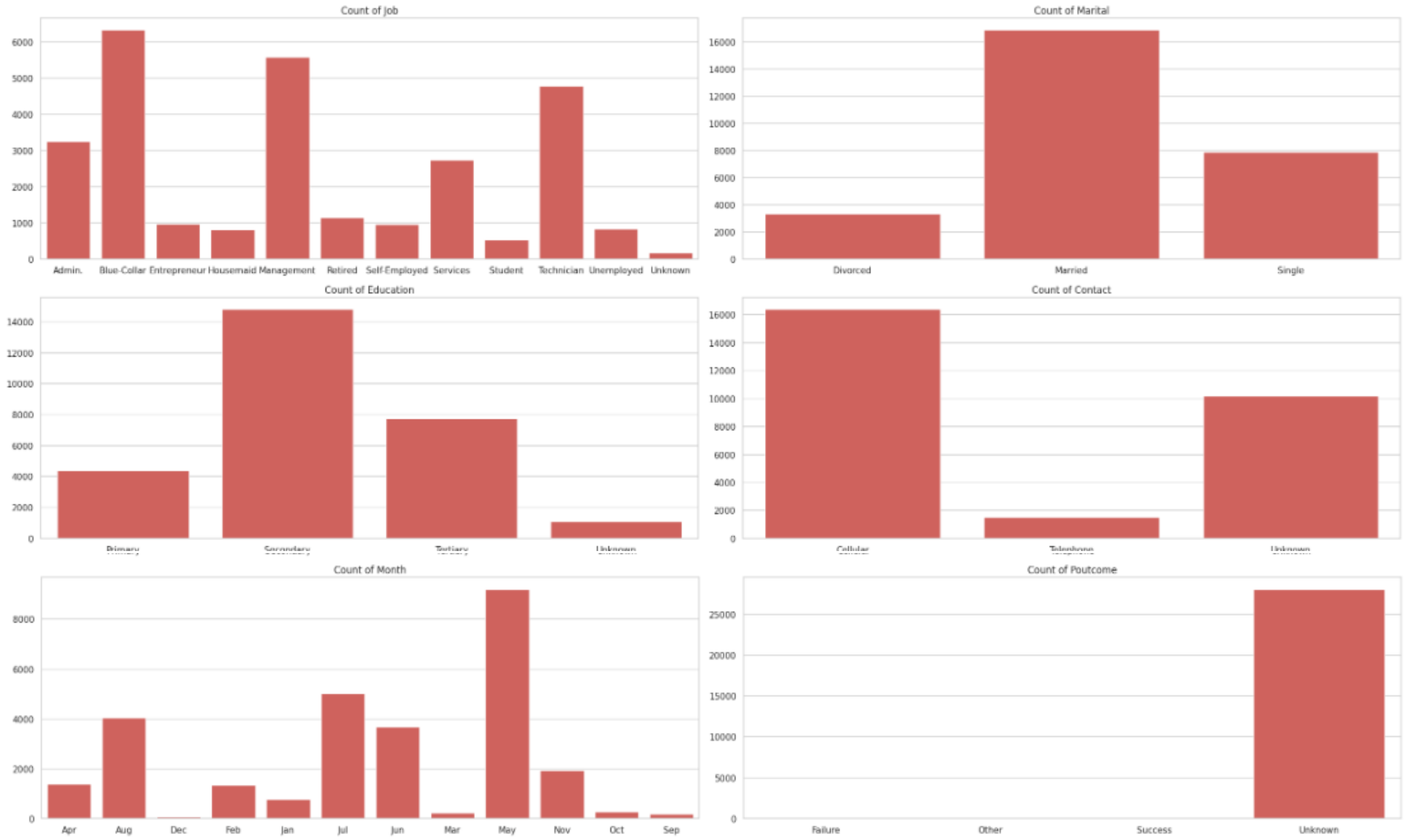
```
[23] num_cols = data.select_dtypes('number').columns.tolist()
     bool_cols = data.select_dtypes(bool).columns.tolist()
     cat_cols = data.select_dtypes('category').columns.tolist()
```

```
sns.countplot(data=data, x='subscribed');
plt.title('Count of Subscribed Term Deposits')
plt.xlabel('')
plt.ylabel('')
plt.show();
```

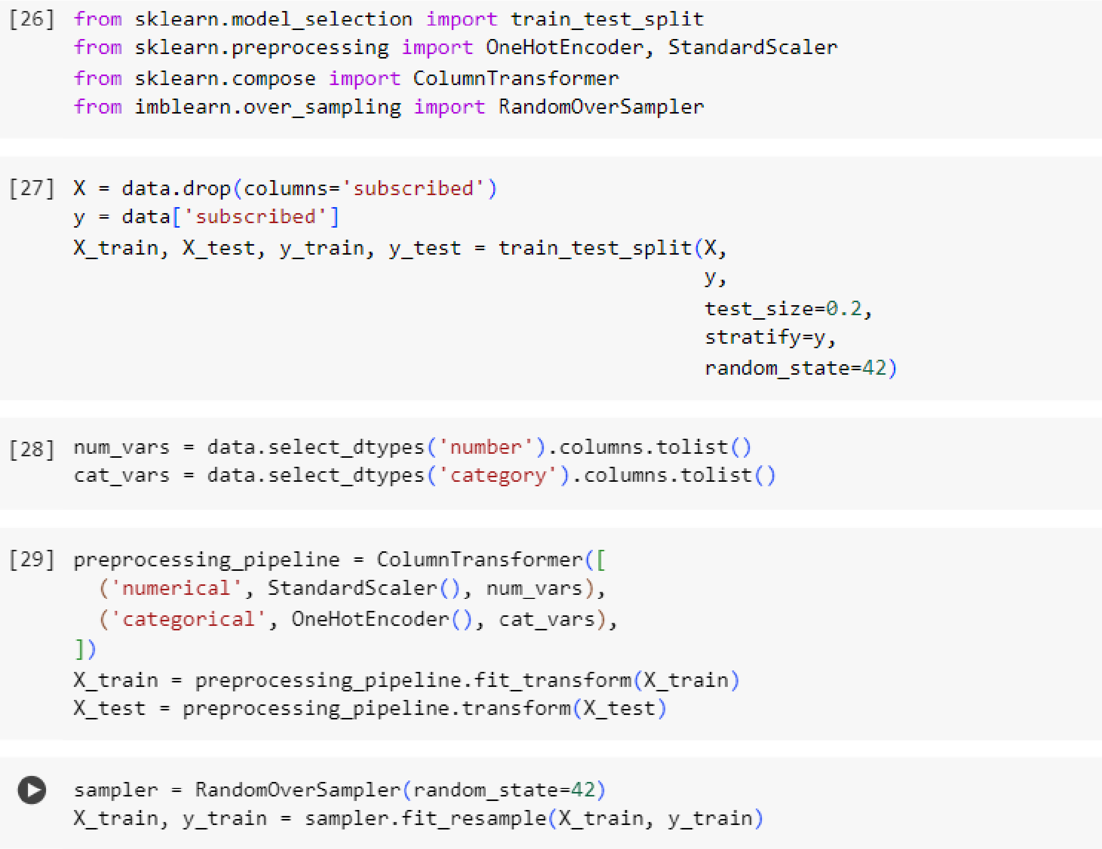




```
fig, axes = plt.subplots(3, 2, figsize=(25, 15))
for feature, ax in zip(cat_cols, axes.flatten()):
  sns.countplot(data=data, x=feature, ax=ax)
  ax.set_title(f'Count of {feature.title()}')
  ax.set_xlabel('')
  ax.set_ylabel('')
plt.tight_layout()
plt.show();
```



Data processing for model training:



```
[26] from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import OneHotEncoder, StandardScaler
     from sklearn.compose import ColumnTransformer
     from imblearn.over_sampling import RandomOverSampler
```

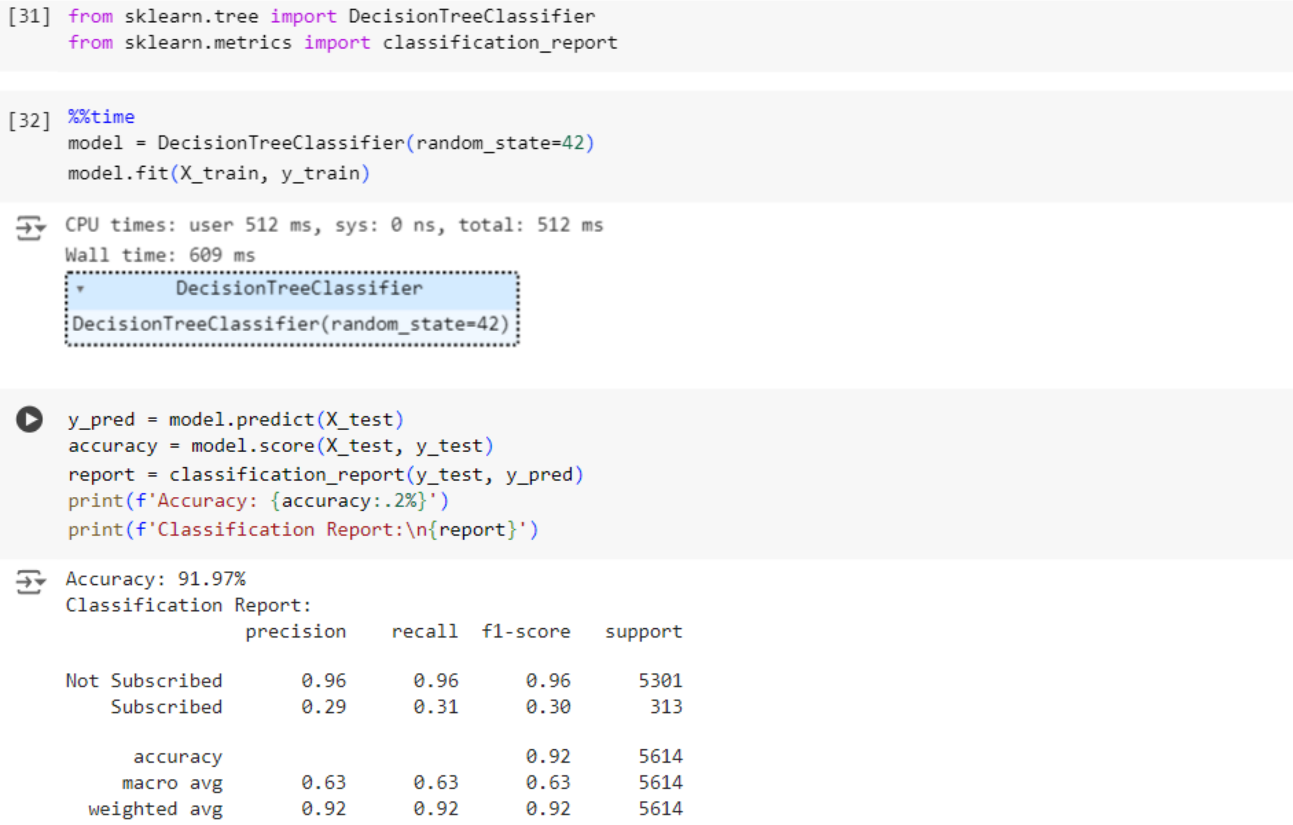
```
[27] X = data.drop(columns='subscribed')
     y = data['subscribed']
     X_train, X_test, y_train, y_test = train_test_split(X,
                                                         y,
                                                         test_size=0.2,
                                                         stratify=y,
                                                         random_state=42)
```

```
[28] num_vars = data.select_dtypes('number').columns.tolist()
     cat_vars = data.select_dtypes('category').columns.tolist()
```

```
[29] preprocessing_pipeline = ColumnTransformer([
       ('numerical', StandardScaler(), num_vars),
       ('categorical', OneHotEncoder(), cat_vars),
     ])
     X_train = preprocessing_pipeline.fit_transform(X_train)
     X_test = preprocessing_pipeline.transform(X_test)
```

```
     sampler = RandomOverSampler(random_state=42)
     X_train, y_train = sampler.fit_resample(X_train, y_train)
```

Building basic model:



```
[31] from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import classification_report
```

```
[32] %%time
     model = DecisionTreeClassifier(random_state=42)
     model.fit(X_train, y_train)
```

```
CPU times: user 512 ms, sys: 0 ns, total: 512 ms
Wall time: 609 ms
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

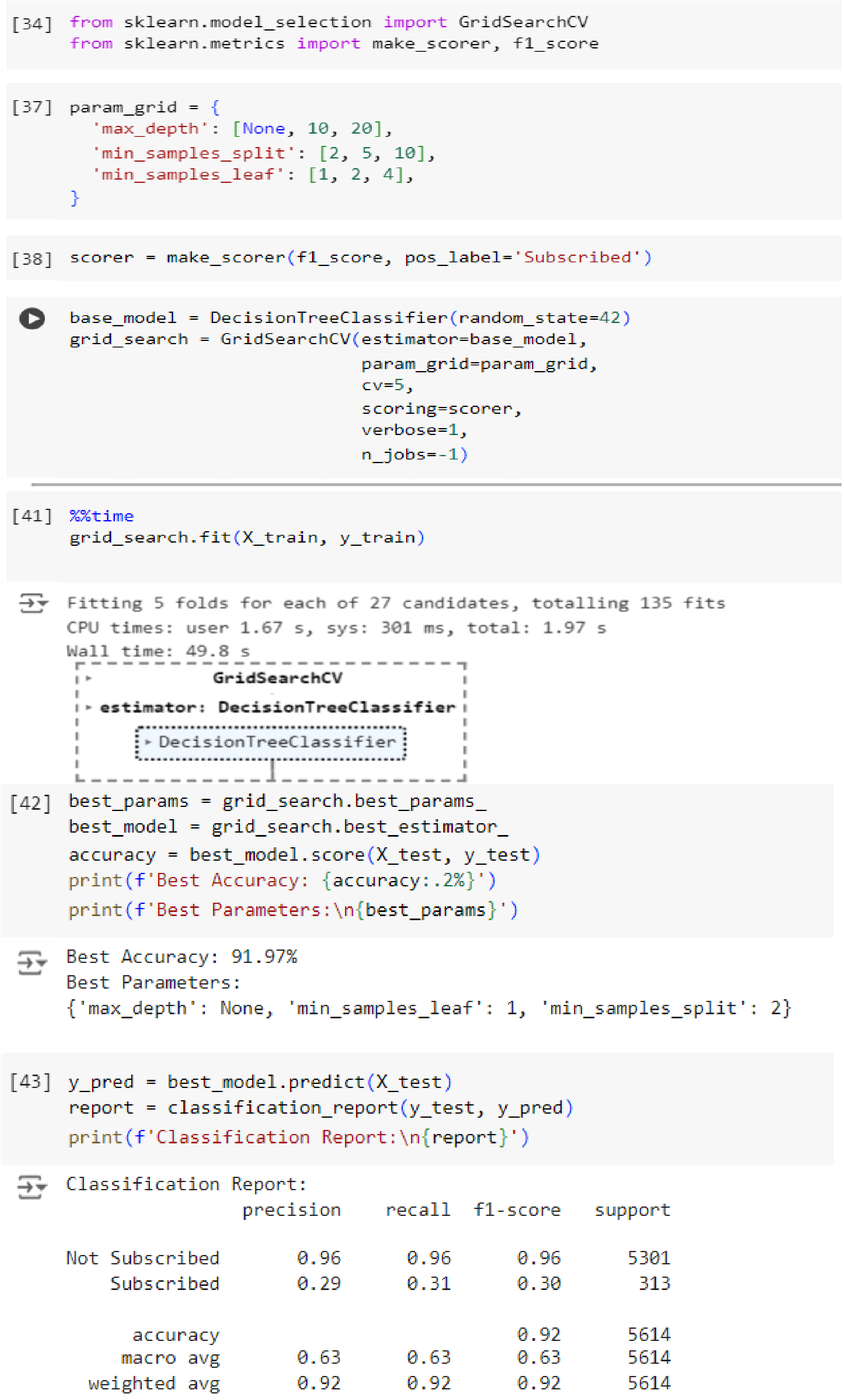
```
y_pred = model.predict(X_test)
accuracy = model.score(X_test, y_test)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy:.2%}')
print(f'Classification Report:\n{report}')
```

```
Accuracy: 91.97%
Classification Report:
                precision    recall  f1-score   support

Not Subscribed       0.96      0.96      0.96      5301
    Subscribed       0.29      0.31      0.30       313

      accuracy                           0.92      5614
     macro avg       0.63      0.63      0.63      5614
  weighted avg       0.92      0.92      0.92      5614
```

Fine tuning the model:



```
[34] from sklearn.model_selection import GridSearchCV
     from sklearn.metrics import make_scorer, f1_score
```

```
[37] param_grid = {
       'max_depth': [None, 10, 20],
       'min_samples_split': [2, 5, 10],
       'min_samples_leaf': [1, 2, 4],
     }
```

```
[38] scorer = make_scorer(f1_score, pos_label='Subscribed')
```

```
     base_model = DecisionTreeClassifier(random_state=42)
     grid_search = GridSearchCV(estimator=base_model,
                                param_grid=param_grid,
                                cv=5,
                                scoring=scorer,
                                verbose=1,
                                n_jobs=-1)
```

```
[41] %%time
     grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
CPU times: user 1.67 s, sys: 301 ms, total: 1.97 s
Wall time: 49.8 s
              GridSearchCV
 ▸ estimator: DecisionTreeClassifier
        ▸ DecisionTreeClassifier
```

```
[42] best_params = grid_search.best_params_
     best_model = grid_search.best_estimator_
     accuracy = best_model.score(X_test, y_test)
     print(f'Best Accuracy: {accuracy:.2%}')
     print(f'Best Parameters:\n{best_params}')
```

```
Best Accuracy: 91.97%
Best Parameters:
{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

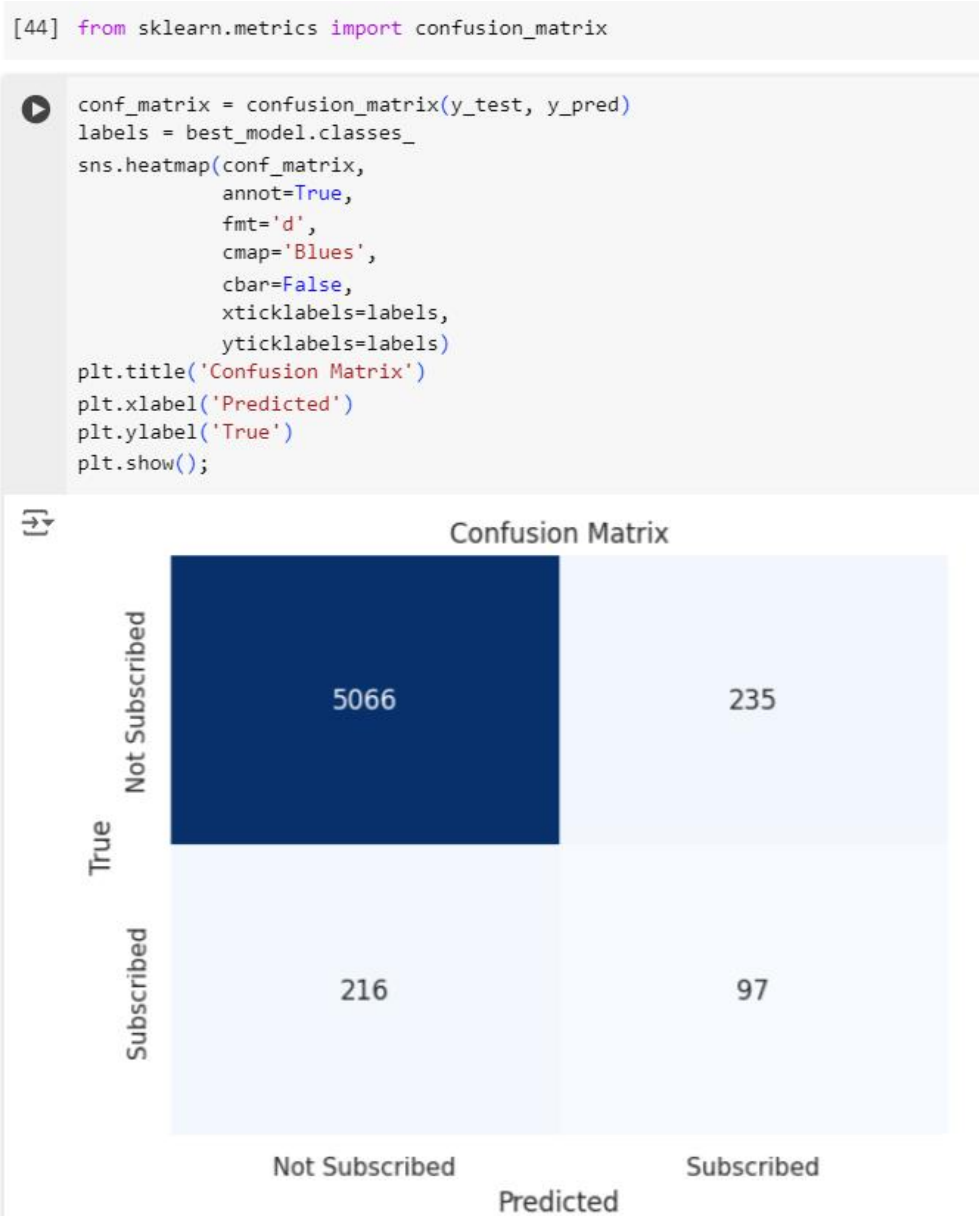
```
[43] y_pred = best_model.predict(X_test)
     report = classification_report(y_test, y_pred)
     print(f'Classification Report:\n{report}')
```

```
Classification Report:
                precision    recall  f1-score   support

Not Subscribed       0.96      0.96      0.96      5301
    Subscribed       0.29      0.31      0.30       313

      accuracy                           0.92      5614
     macro avg       0.63      0.63      0.63      5614
  weighted avg       0.92      0.92      0.92      5614
```

Testing the Result:



```
[44] from sklearn.metrics import confusion_matrix
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
labels = best_model.classes_
sns.heatmap(conf_matrix,
            annot=True,
            fmt='d',
            cmap='Blues',
            cbar=False,
            xticklabels=labels,
            yticklabels=labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show();
```