Data science Internshala Training Project

Priyangshu Sarkar

Problem Statement:- Your client is a retail banking institution. Term deposits are a major source of income for a bank. A term deposit is a cash investment held at a financial institution. Your money is invested for an agreed rate of interest over a fixed amount of time, or term. The bank has various outreach plans to sell term deposits to their customers such as email marketing, advertisements, telephonic marketing and digital marketing. Telephonic marketing campaigns still remain one of the most effective way to reach out to people. However, they require huge investment as large call centers are hired to actually execute these campaigns. Hence, it is crucial to identify the customers most likely to convert beforehand so that they can be specifically targeted via call. You are provided with the client data such as : age of the client, their job type, their marital status, etc. Along with the client data, you are also provided with the information of the call such as the duration of the call, day and month of the call, etc. Given this information, your task is to predict if the client will subscribe to term deposit.

--importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Classification

```
train = pd.read_csv('train[1].csv')
test = pd.read_csv('test[1].csv')
```

```
train
```

	ID	age	job	marital	education	default	balance	housing	loan
0	26110	56	admin.	married	unknown	no	1933	no	no
1	40576	31	unknown	married	secondary	no	3	no	no
2	15320	27	services	married	secondary	no	891	yes	no
3	43962	57	management	divorced	tertiary	no	3287	no	no
4	29842	31	technician	married	secondary	no	119	yes	no
...	...	...	...	...	...	...	...	...	...
31642	36483	29	management	single	tertiary	no	0	yes	no

test

	ID	age	job	marital	education	default	balance	housing	loan
0	38441	32	services	married	secondary	no	118	yes	no
1	40403	78	retired	divorced	primary	no	2787	no	no
2	3709	31	self-employed	single	tertiary	no	144	yes	no
3	37422	57	services	single	primary	no	3777	yes	no
4	12527	45	blue-collar	divorced	secondary	no	-705	no	yes
...	...	...	...	...	...	...	...	...	...
13559	23465	39	management	married	tertiary	no	45	no	no
13560	11743	54	blue-collar	married	primary	no	2281	yes	no
13561	28292	35	retired	married	primary	no	285	yes	no
13562	45163	29	admin.	single	secondary	no	464	no	no
13563	34839	29	admin.	married	secondary	no	2	yes	no

13564 rows × 17 columns

```
train.columns
```

```
Index(['ID', 'age', 'job', 'marital', 'education', 'default', 'balance',
       'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign',
       'pdays', 'previous', 'poutcome', 'subscribed'],
      dtype='object')
```

```
test.columns
```

```
Index(['ID', 'age', 'job', 'marital', 'education', 'default', 'balance',
       'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign',
       'pdays', 'previous', 'poutcome'],
      dtype='object')
```

```
train.shape
```

```
(31647, 18)
```

```
test.shape
```

```
(13564, 17)
```

```
train.head()
```

	ID	age	job	marital	education	default	balance	housing	loan	con
0	26110	56	admin.	married	unknown	no	1933	no	no	telepl
1	40576	31	unknown	married	secondary	no	3	no	no	ce
2	15320	27	services	married	secondary	no	891	yes	no	ce
3	43962	57	management	divorced	tertiary	no	3287	no	no	ce
4	29842	31	technician	married	secondary	no	119	yes	no	ce

Univarient Analysis

```
train['subscribed'].value_counts()
```

```
no     27932
yes     3715
Name: subscribed, dtype: int64
```

```
train['subscribed'].value_counts(normalize=True)
```

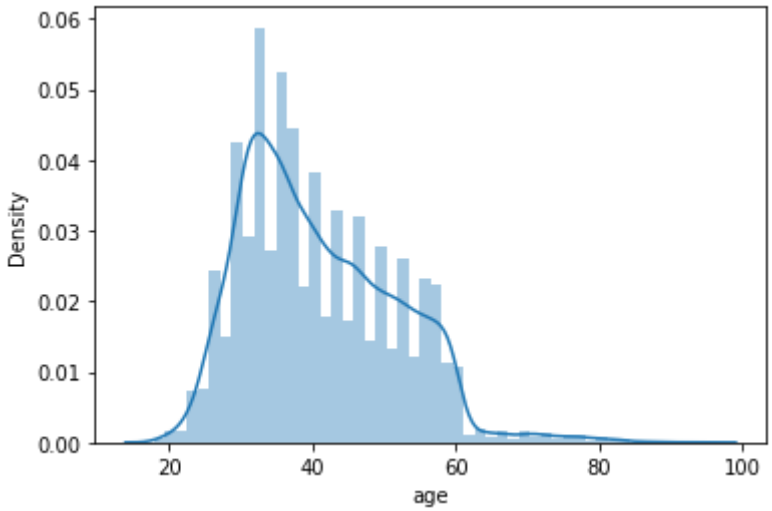
```
no     0.882611
yes    0.117389
Name: subscribed, dtype: float64
```

```
train['subscribed'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f60915d8110>
```

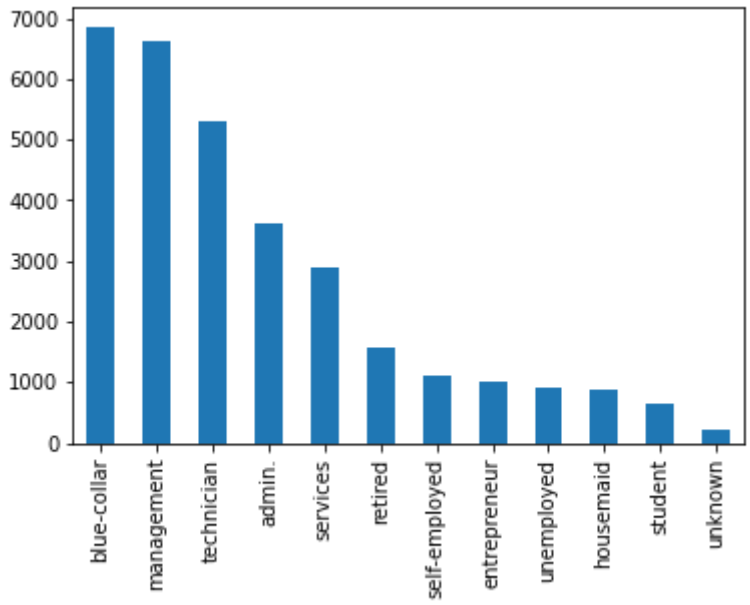
```
sn.distplot(train["age"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f609151ed50>
```




```
train['job'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6091090e90>
```




```
train['default'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6090fe0090>
```

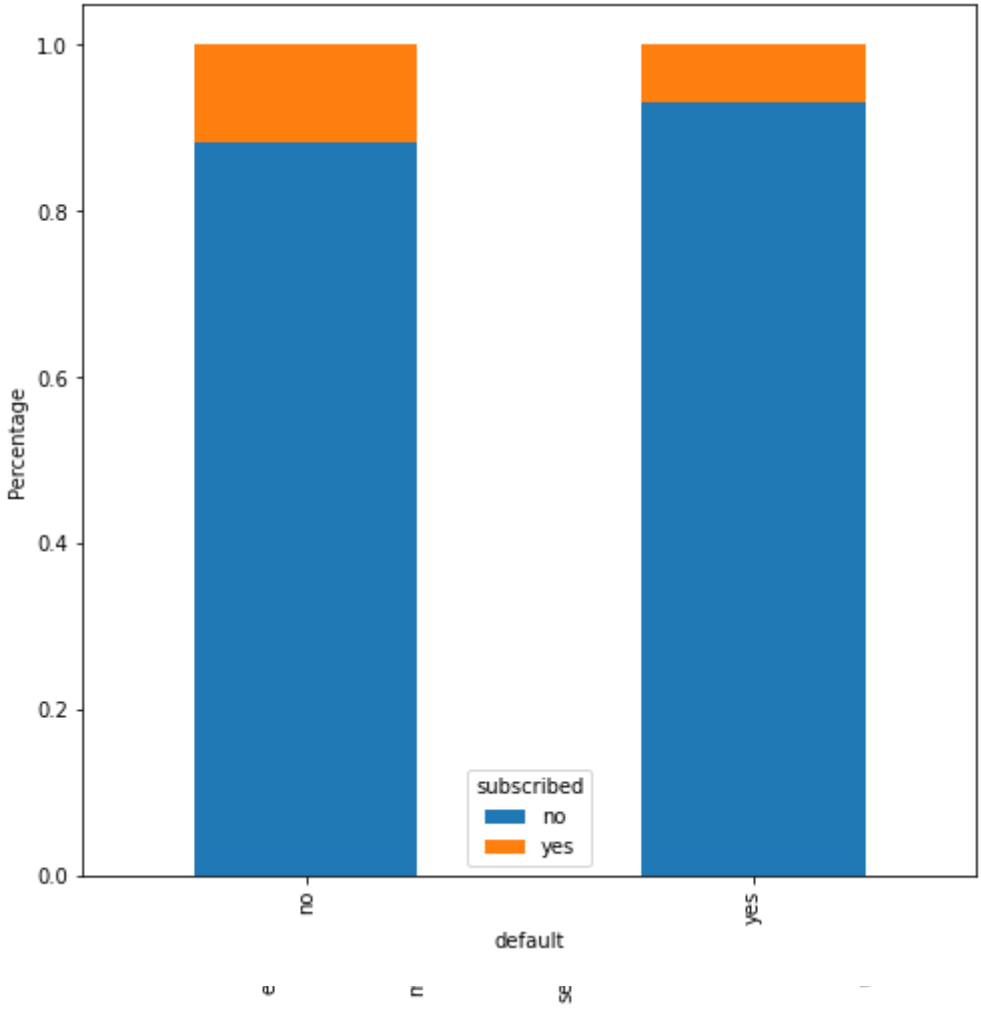
Bivarient Analysis

```
print(pd.crosstab(train['job'],train['subscribed']))

job=pd.crosstab(train['job'],train['subscribed'])
job.div(job.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True,figsize=(8,8))
plt.xlabel('Job')
plt.ylabel('Percentage')
```

```
subscribed       no  yes
job
admin          3179  452
```

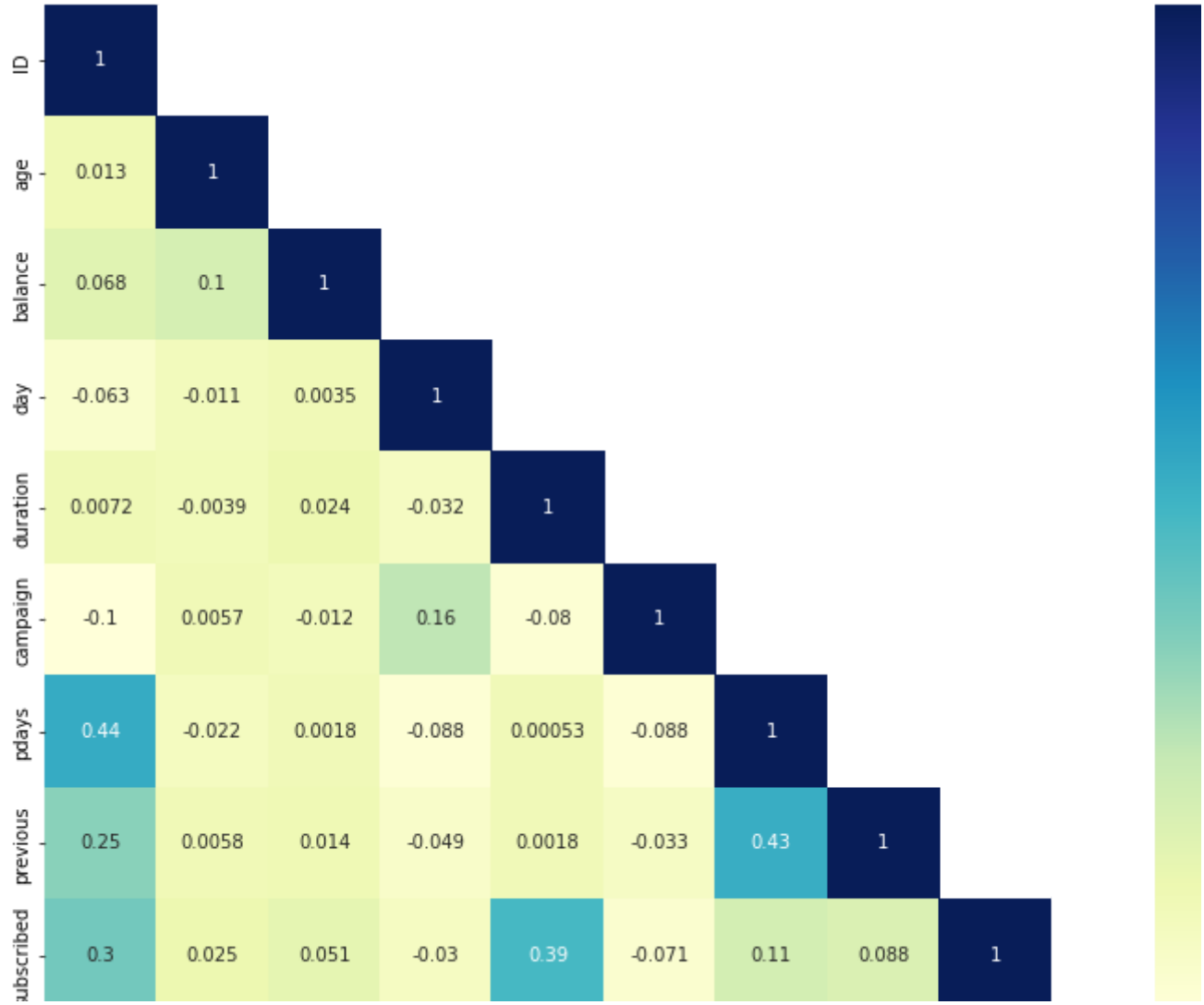
```
print(pd.crosstab(train['default'],train['subscribed']))

default=pd.crosstab(train['default'],train['subscribed'])
default.div(default.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(
plt.xlabel('default')
plt.ylabel('Percentage')
```

```
subscribed     no   yes
default
no          27388  3674
yes           544    41
Text(0, 0.5, 'Percentage')
```




```
train['subscribed'].replace('no', 0,inplace=True)
train['subscribed'].replace('yes', 1,inplace=True)


corr = train.corr()
mask = np.array(corr)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sn.heatmap(corr, mask=mask,vmax=.9, square=True,annot=True, cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6090fbe250>
```




```
train.isnull().sum()
```

```
ID            0
age           0
job           0
marital       0
education     0
default       0
balance       0
housing       0
loan          0
contact       0
day           0
month         0
duration      0
campaign      0
pdays         0
previous      0
poutcome      0
subscribed    0
dtype: int64
```

```
target = train['subscribed']
train = train.drop('subscribed',1)
```

```
train = pd.get_dummies(train)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_val, y_train, y_val = train_test_split(train, target, test_size = 0.2, random_s
```

Regression

```
from sklearn.linear_model import LogisticRegression
```

```
lreg = LogisticRegression()
```

```
lreg.fit(X_train,y_train)
```

```
     LogisticRegression()
```

```
prediction = lreg.predict(X_val)
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_val, prediction)
```

```
     0.8913112164296998
```

Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf = DecisionTreeClassifier(max_depth=4, random_state=0)
```

```
clf.fit(X_train,y_train)
```

```
     DecisionTreeClassifier(max_depth=4, random_state=0)
```

```
predict = clf.predict(X_val)
```

```
accuracy_score(y_val, predict)
```

```
     0.9042654028436019
```

```
test = pd.get_dummies(test)
```

```
test_prediction = clf.predict(test)


submission = pd.DataFrame()


submission['ID'] = test['ID']
submission['subscribed'] = test_prediction


submission['subscribed'].replace(0,'no',inplace=True)
submission['subscribed'].replace(1,'yes',inplace=True)


submission.to_csv('submission.csv', header=True, index=False)
```