In [ ]:

```
import pandas as pd
import numpy as np                         # For mathematical calculations
import seaborn as sns                      # For data visualization
import matplotlib.pyplot as plt
import seaborn as sn                       # For plotting graphs
%matplotlib inline
import warnings                            # To ignore any warnings
warnings.filterwarnings("ignore")
```

In [8]:

```
train = pd.read_csv(r"C:\Users\B Akhil\OneDrive\Desktop\DataScience\train.csv") #rea
#print(train)
test = pd.read_csv(r'C:\Users\B Akhil\OneDrive\Desktop\DataScience\test.csv') #readi
#print('test')
```

In [25]:

```
train.columns
```

Out[25]:

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

In [26]:

```
train.shape, test.shape #( No. of rows, No. of columns )
```

Out[26]:

```
((31647, 18), (13564, 17))
```

In [27]:

```
train.dtypes
```

Out[27]:

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

In [13]:

```
#printing first five rows of the dataset
train.head()

#printing last five rows of the dataset
#train.tail()
```

Out[13]:

	ID	age	job	marital	education	default	balance	housing	loan	contact	day
0	26110	56	admin.	married	unknown	no	1933	no	no	telephone	19
1	40576	31	unknown	married	secondary	no	3	no	no	cellular	20
2	15320	27	services	married	secondary	no	891	yes	no	cellular	18
3	43962	57	management	divorced	tertiary	no	3287	no	no	cellular	22
4	29842	31	technician	married	secondary	no	119	yes	no	cellular	4

In [29]:

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

Out[29]:

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

In [30]:

```
# Normalize can be set to True to print proportions instead of number

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

Out[30]:

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

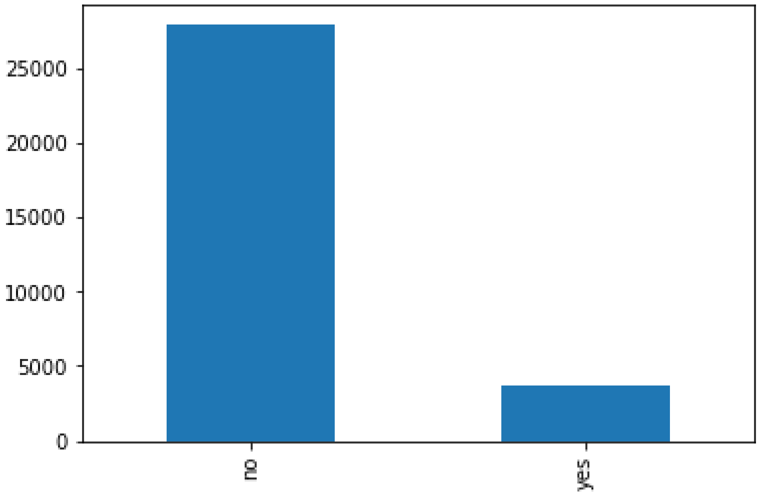
In [31]:

```
# plotting the bar plot of frequencies

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

Out[31]:

```
<AxesSubplot:>
```

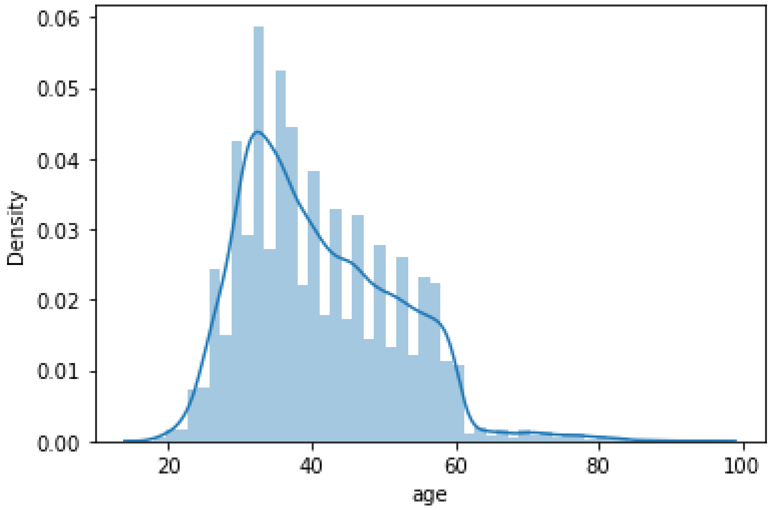


In [32]:

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

Out[32]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```

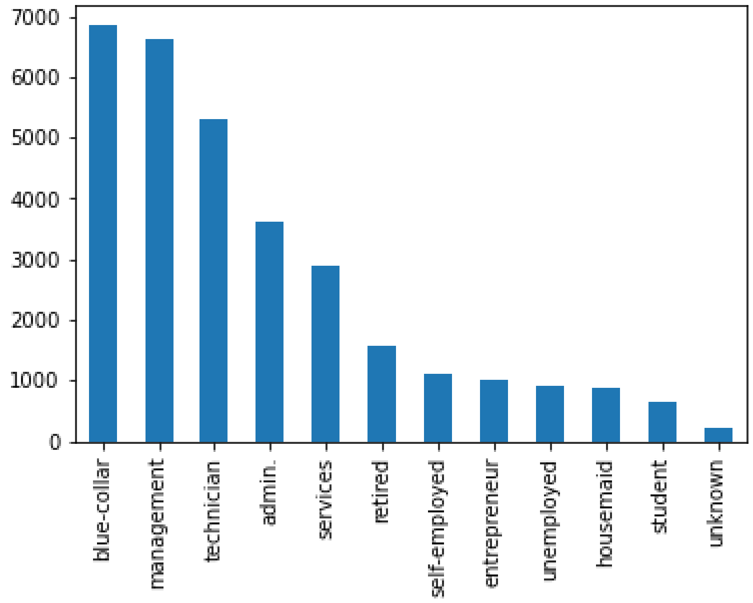


In [33]:

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

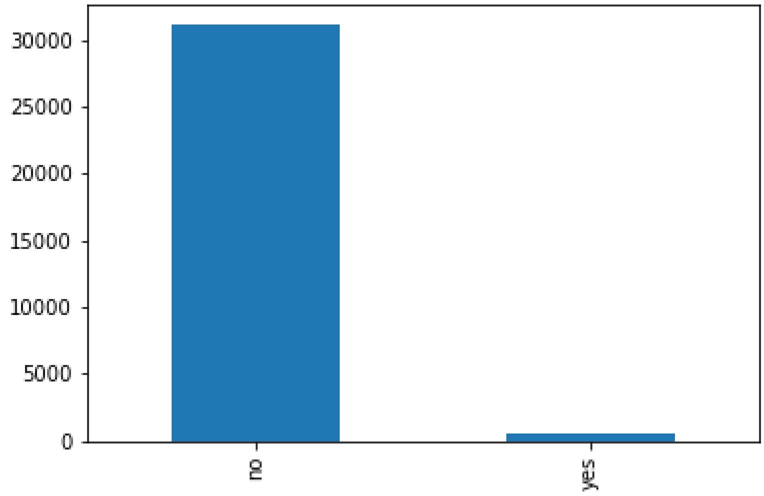
Out[33]:

```
<AxesSubplot:>
```



In [34]:
```
train['default'].value_counts().plot.bar()
```

Out[34]: `<AxesSubplot:>`

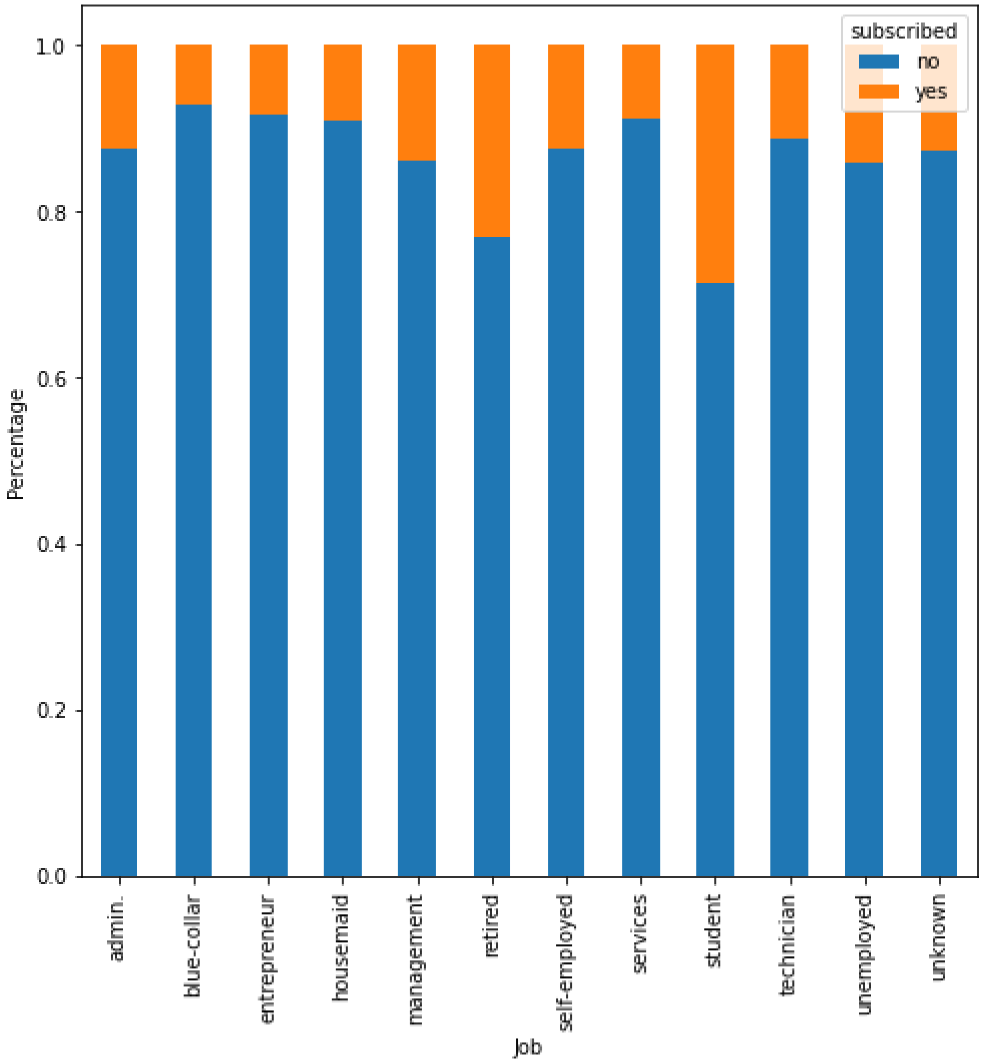


In [35]:
```
print(pd.crosstab(train['job'],train['subscribed']))

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

```
subscribed       no  yes
job
admin.         3179  452
blue-collar    6353  489
entrepreneur    923   85
housemaid       795   79
management     5716  923
retired        1212  362
self-employed   983  140
services       2649  254
student         453  182
technician     4713  594
unemployed      776  129
unknown         180   26
```

Out[35]: `Text(0, 0.5, 'Percentage')`



In [36]:

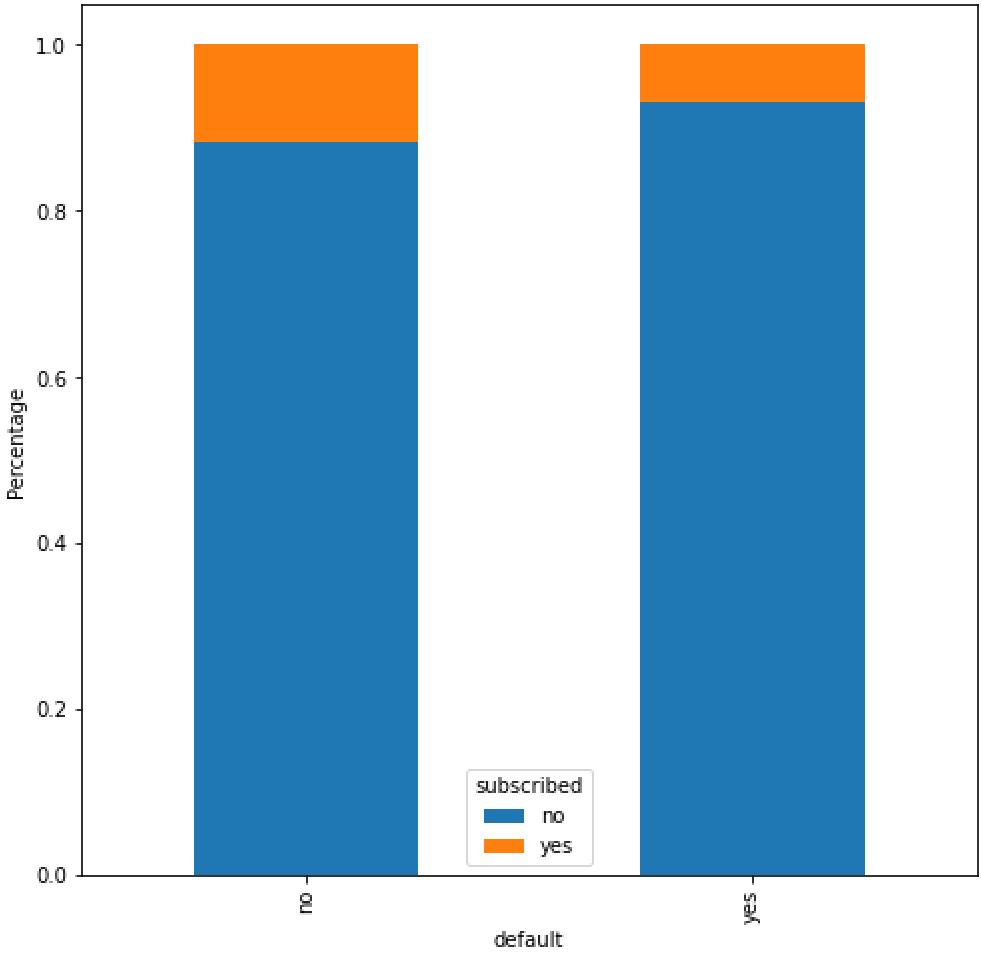
```
print(pd.crosstab(train['default'],train['subscribed']))

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

```
subscribed     no   yes
default
no          27388  3674
yes           544    41
```

Out[36]:

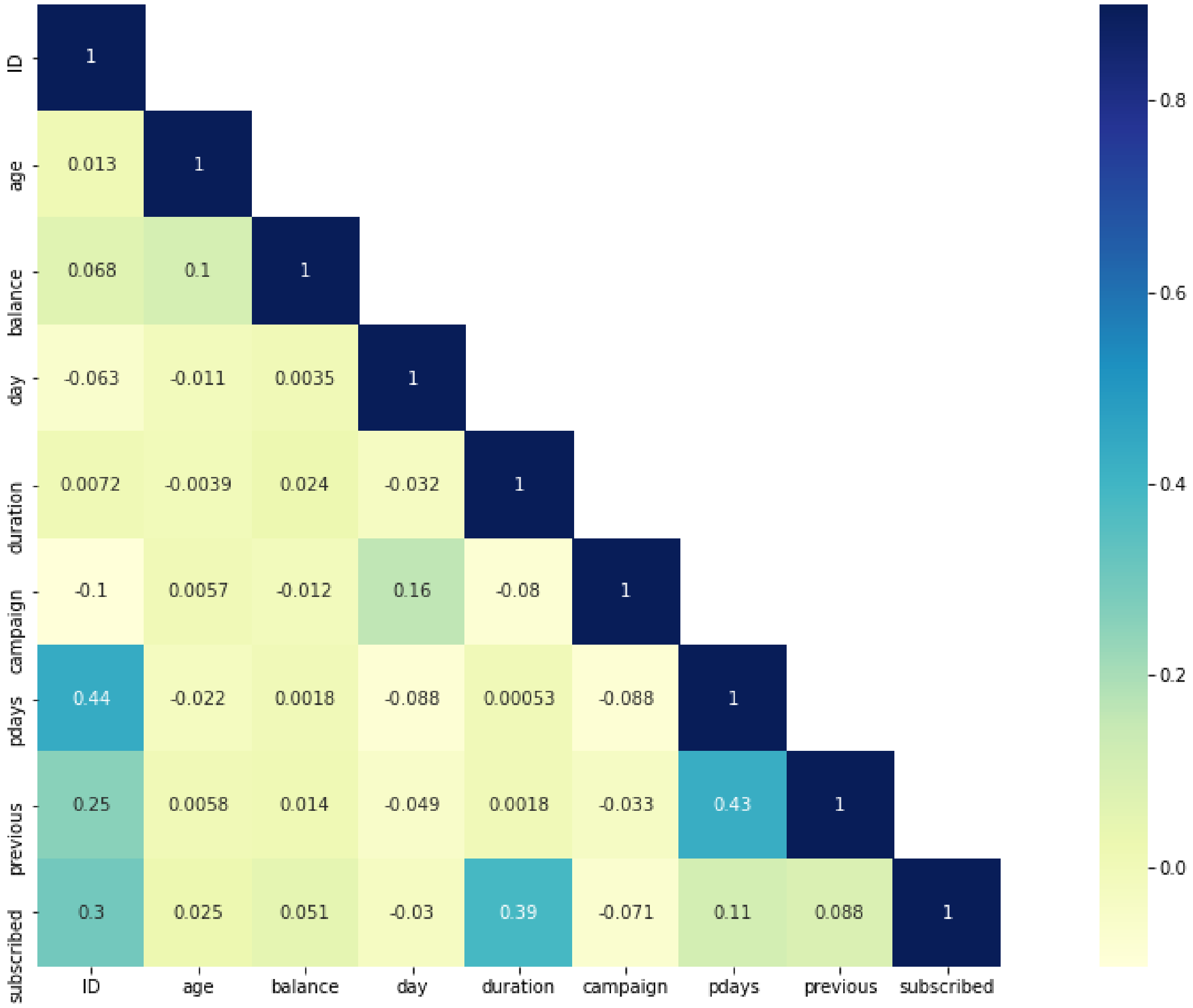
```
Text(0, 0.5, 'Percentage')
```



```
In [37]: train['subscribed'].replace('no', 0,inplace=True)
         train['subscribed'].replace('yes', 1,inplace=True)
```

```
In [38]: 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")
```

```
Out[38]: <AxesSubplot:>
```



In [39]:

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

Out[39]:

```
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
```

In [40]:

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

In [41]:

```
# applying dummies on the train dataset
train = pd.get_dummies(train)
```

In [42]:

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

In [43]:

```
# splitting into train and validation with 20% data in validation set and 80% data i
```

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

In [44]:

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

In [45]:

```
# defining the logistic regression model
lreg = LogisticRegression()
```

In [46]:

```
# fitting the model on  X_train and y_train
lreg.fit(X_train,y_train)
```

Out[46]:

```
LogisticRegression()
```

In [47]:

```
# making prediction on the validation set
prediction = lreg.predict(X_val)
```

In [48]:

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

In [49]:

```
# calculating the accuracy score
accuracy_score(y_val, prediction)
```

Out[49]:

```
0.8913112164296998
```

In [50]:

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

In [51]:

```
# defining the decision tree model with depth of 4, you can tune it further to impro
clf = DecisionTreeClassifier(max_depth=4, random_state=0)
```

In [52]:

```
# fitting the decision tree model
clf.fit(X_train,y_train)
```

Out[52]:

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

In [53]:

```
# making prediction on the validation set
predict = clf.predict(X_val)
```

In [54]:

```
# calculating the accuracy score
accuracy_score(y_val, predict)
```

Out[54]:

```
0.9042654028436019
```

In [55]:

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

In [56]:

```
test_prediction = clf.predict(test)
```

In [57]:

```
submission = pd.DataFrame()
```

In [58]:

```
# creating a Business_Sourced column and saving the predictions in it
submission['ID'] = test['ID']
submission['subscribed'] = test_prediction
```

In [59]:

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

In [62]:

```
submission.to_csv(r'C:\Users\B Akhil\OneDrive\Desktop\DataScience\submission.csv', h
```

In [ ]:

In [ ]: