1. INTRODUCTION

1.1 Overview:

It is important for banks to optimize the marketing strategies and improve effectiveness. Understanding customer needs leads to effective marketing plans and greater customer satisfaction.
In this project we will enable the bank to develop a more understanding of it's customer base data and predicts the customer's response and also creates a customer profile for future marketing based on the data provided.

1.2 Purpose:

From the given data, analyzing the customer base such as age, loan, Poutcomes, housing, job etc., the bank will be able to predict the customer behaviors and will be able to predict which customer is more likely to make term deposit so that the bank can focus more on those customers.

2. LITERATURE SURVEY

2.1 Existing Problem:

The challenges faced by the retail banking :

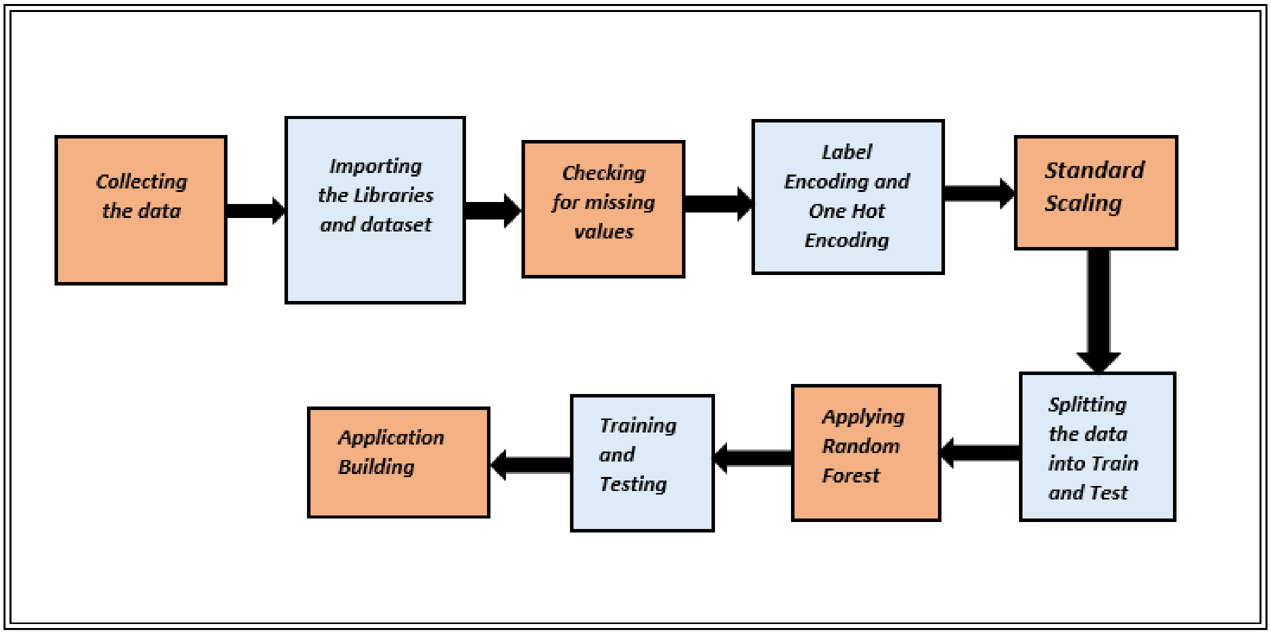
- Marketing spending in bank is massive.
- It is important to optimize the marketing strategies.
- Lack of customer's interaction leading to not knowing the future term deposits.
- Investing in new markets.
- Finding new profit opportunities.
- Decision making support.

2.2 Proposed Solution:

- By using simple open software methods we can build a model to observe the customer's features.
- This model helps in predicting whether the customer will make a term deposit or not.
- Using the real data by training the model with proper algorithm this can be achieved.
- This can be done in simple way and also helps in improving company's performance.

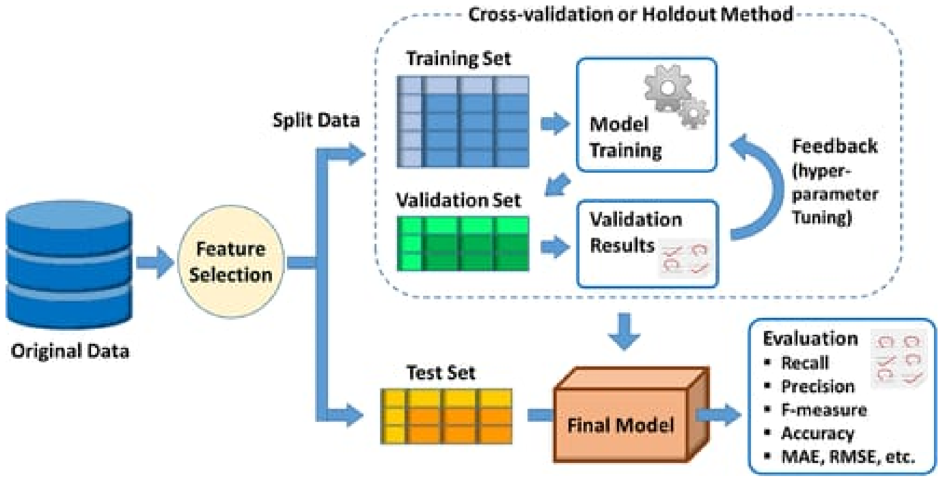
3. THEORITICAL ANALYSIS

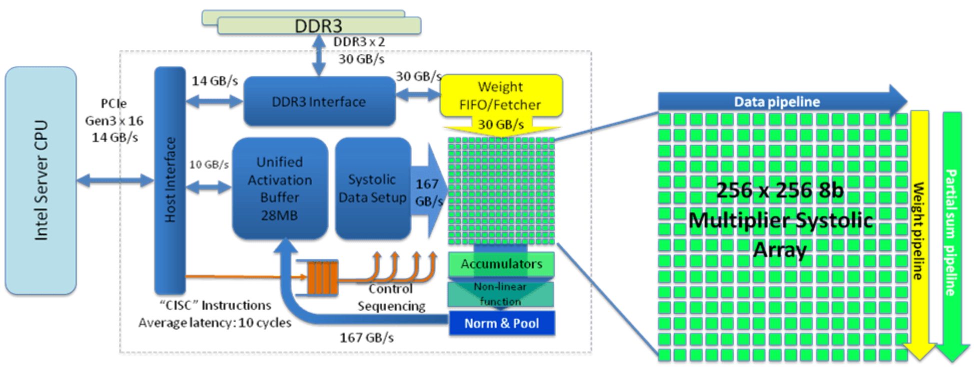
3.1 Block Diagram :



3.2 Hardware and Software Designing:

Software designing involves in envisioning and defining software solutions to one or more sets of problems . Software designing of retail banking is based on understanding the customers, creating and marketing products that directly address their needs. Every design has the different way to approach for a particular given problem solving statements.



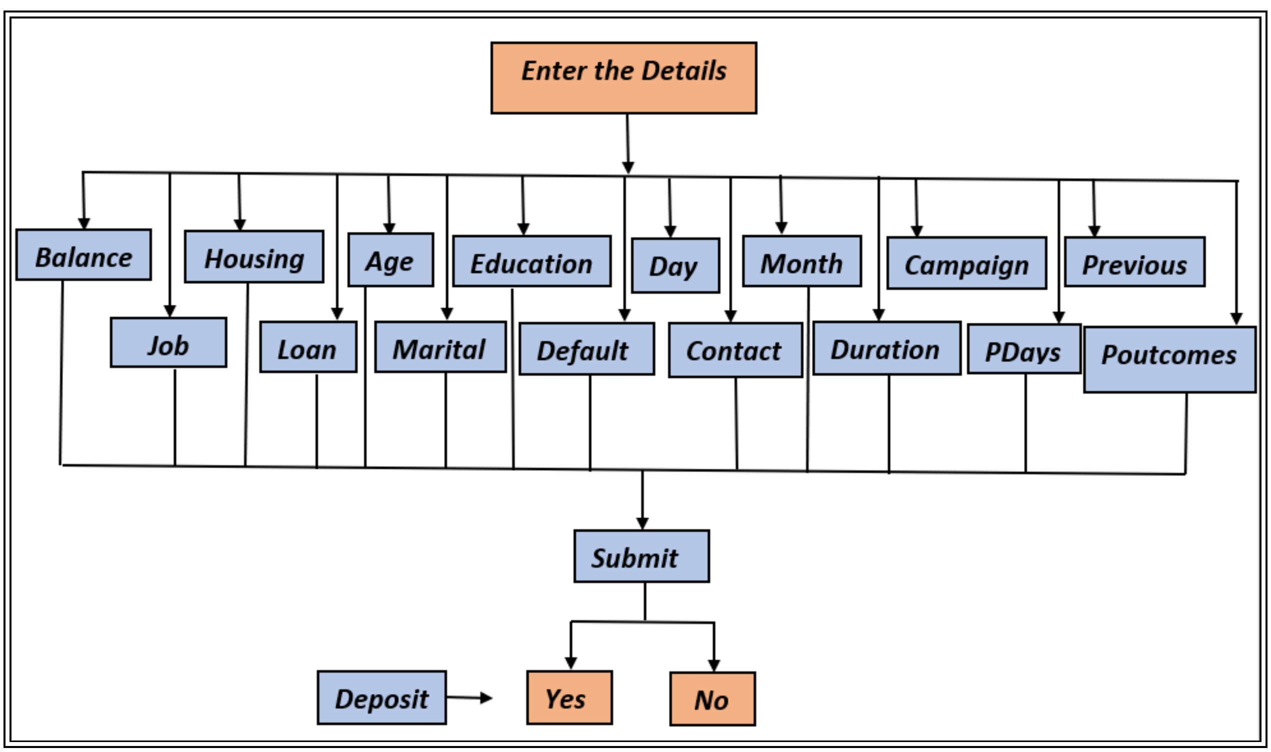


4. EXPERIMENTAL INVESTIGATIONS

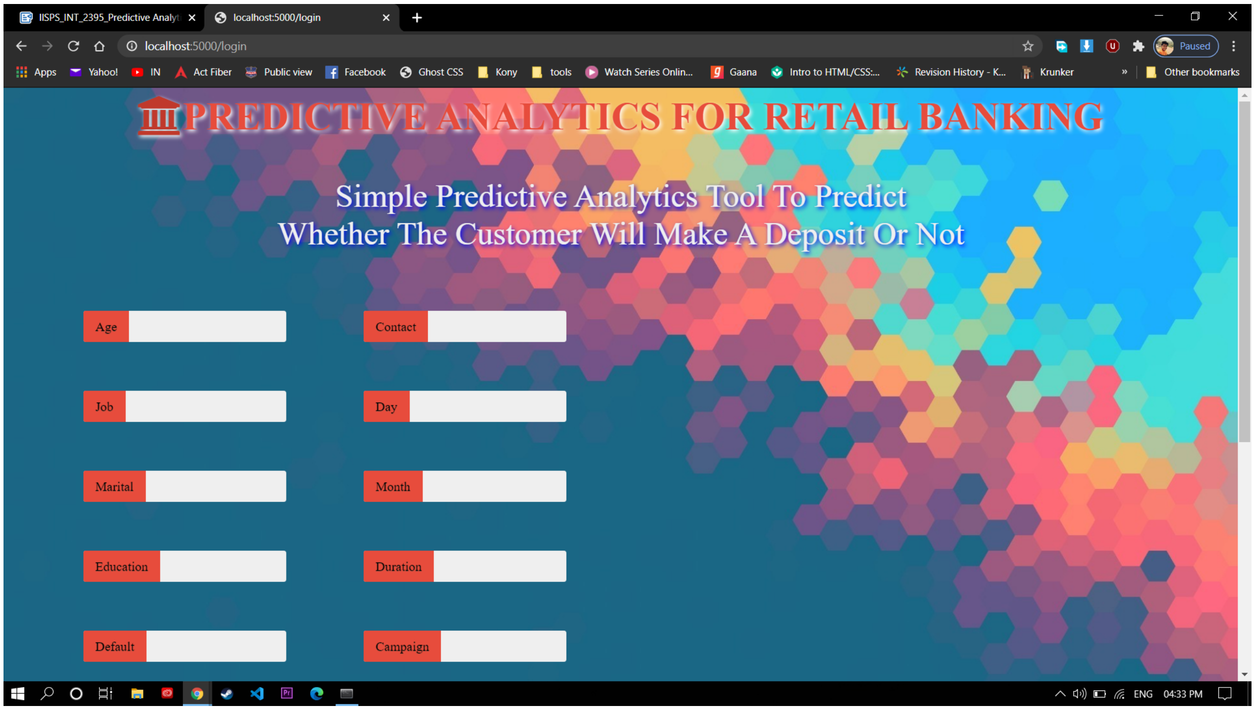
The focus is of on Predictive Analytics of Retail Banking. To compare the performances of the model by using different algorithms like Decision Tree, Random Forest, Naive Baye's. The accuracy for the algorithms Decision Tree, Naive Baye's was low. Decision Tree gave an accuracy of 78% and Naive Baye's algorithm gave an accuracy of 72%. Random Forest gave an

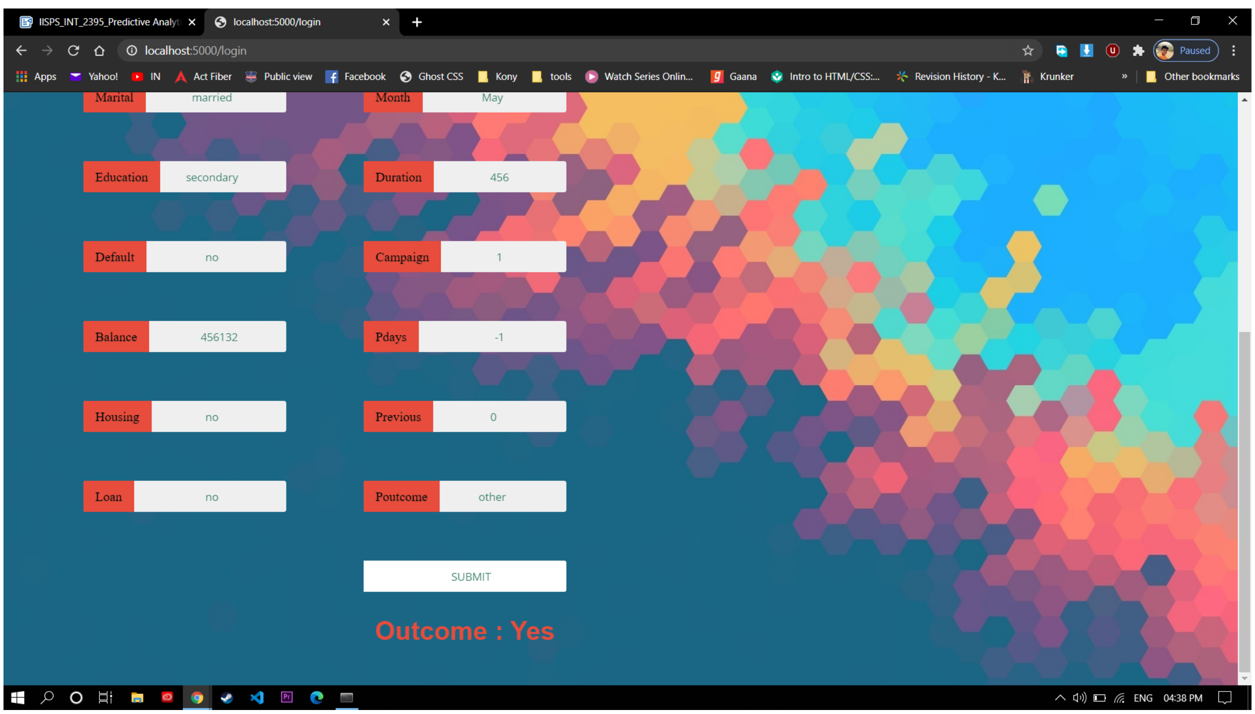
accuracy of 86% and the ROC_AUC of this is 0.86.Almost every feature is important in predicting the term deposit.
The complete training of the model predicts whether the customer will make the term deposit or not.
The visualization of age and deposit gave a linearly increasing line which shows that customers from the age 30-55 made more deposits than the customers below 30. Visualization of Balance and Deposit also gave linearly increasing line. More customers of the marital status "married" made more deposits.

5. FLOWCHART



6. RESULT


localhost:5000/login
PREDICTIVE ANALYTICS FOR RETAIL BANKING
Simple Predictive Analytics Tool To Predict
Whether The Customer Will Make A Deposit Or Not
Age
Contact
Job
Day
Marital
Month
Education
Duration
Default
Campaign


localhost:5000/login
Marital
married
Month
May
Education
secondary
Duration
456
Default
no
Campaign
1
Balance
456132
Pdays
-1
Housing
no
Previous
0
Loan
no
Poutcome
other
SUBMIT
Outcome : Yes

7. ADVANTAGES AND DISADVANTAGES

Advantages :

- Time is saved. Instead of checking the massive data set for predicting it is easy to predict using this model.
- It is easy to identify regular depositors so that more benefits can be given.Marketing strategies and effectiveness.
- Understanding the customer's base for greater satisfaction.
- Marketing spending by the bank will be reduced.
- Helps in taking decisions.
- Interaction with customer increases.
- Immediate Response & Support.
- Customer's behavior recognition.

Disadvantages :

- Accuracy deficiency.
- Collection of data (Massive Data).
- Marketing the Web Application is not such easy.

8. APPLICATIONS

- This is used in banking sector for predictions.
- Predicting the amount customer will deposit so that banks can have future scope.

9. Conclusion

A simple predictive analytics model can be carried out on real data using open source statistical modeling software. The results can be applied to produce real tangible improvements in a company's business performance.

10. FUTURE SCOPE

- Predictive analytics to extract actionable insights and quantifiable predictions can help the banks to gain insights that comprise of all types of customer behavior.
- Since this reduces the unnecessary work and saves time which is main factor for every sector, this model can be used.
- With the steady increase in the growing demand for the analytics, which has successfully managed to produce more sophisticated and accurate results, many more banks are deploying a range of analytics today.

11. BIBILIOGRAPHY

Used tools

for Model Building :

- Jupyter Notebook 6.0.3 (anaconda - 3)

for Application Building

- Spyder 4.0.1 (anaconda-3)
- HTML
- CSS

Refference links :

- Data Collection:

https://thesmartbridge.com/documents/spsaimldocs/datasets/bank.csv

- Visualization:

https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7

- Data Preprocessing:

https://thesmartbridge.com/documents/spsaimldocs/Datapreprocessing.pdf

- Model Building:

https://thesmartbridge.com/documents/spsaimldocs/Machinelearning.pdf

- Application Building:

https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp
https://thesmartbridge.com/documents/spsaimldocs/FlaskML.pdf
https://htmlcolorcodes.com/
https://www.w3schools.com/icons/bootstrap_icons_glyphicons.asp
https://wallpaperset.com/wall/eyJpdiI6IlJcL3VPaGdoS09GUkNXSEFyZEFlOEh3PT0iLCJ2YWx1ZSI6IkxKaDJoazAwbkJiSkhzRTFQM28xbnc9PSIsIm1hYyI6IjdhYj

kwNWRhZjE2MTY3Njg0ZGM4ZWIxYTk5MzlkNTU1NGZiN2JmYzM3ODU5NTQ2
ZWEzNGZjZjQyZjg2ZGM0NzAifQ==

12. APPENDIX

Source Code :

Importing The Libraries :

In [1]:
```
import numpy as np
import pandas as pd
```

Importing The Dataset :

In [2]:
```
ds=pd.read_csv(r'bank.csv')
```

In [3]:
```
ds.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

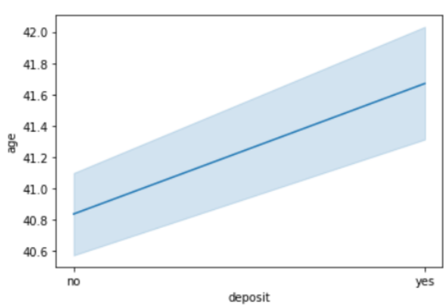
Visualization:

In [4]:
```
import seaborn as sea
import matplotlib.pyplot as plt
%matplotlib inline
```

Visualize the Age with Deposit:

In [5]:
```
sea.lineplot(x="deposit",y="age",data=ds)
```

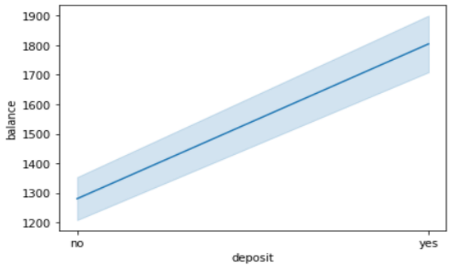
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc027c648>



Visualize Balance with Deposit:

In [6]:
```
sea.lineplot(x="deposit",y="balance",data=ds)
```

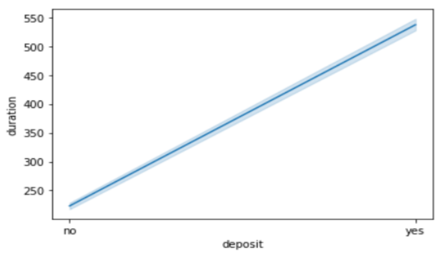
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1bd34c8>



Visualize Duration with Deposit:

In [7]:
```
sea.lineplot(x="deposit",y="duration",data=ds)
```

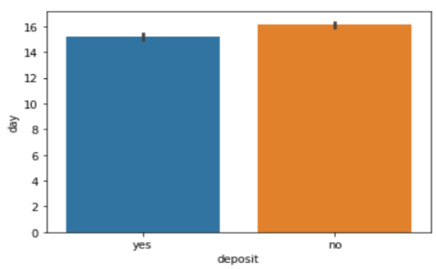
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1c39b48>



Visualize Day with Deposit:

In [8]:
```
sea.barplot(x="deposit",y="day",data=ds)
```

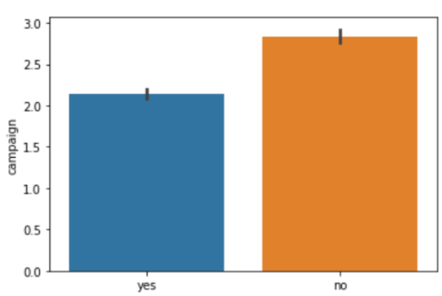
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1c92f88>



Visualize Campaign with Deposit:

In [9]:
```
sea.barplot(x="deposit",y="campaign",data=ds)
```

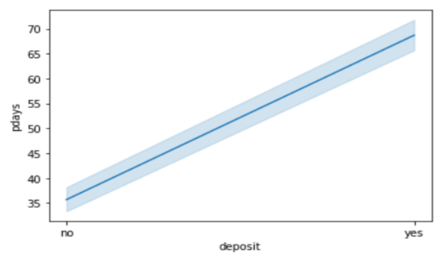
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1d00f08>



Visualize pdays with Deposit:

In [10]:
```
sea.lineplot(x="deposit",y="pdays",data=ds)
```

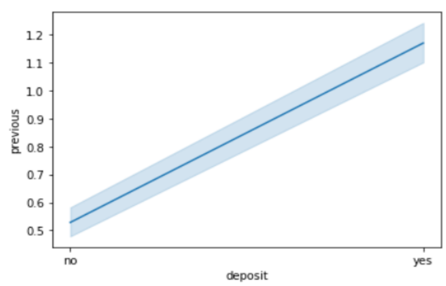
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1d74b08>



Visualize Previous with Deposit:

In [11]:
```
sea.lineplot(x="deposit",y="previous",data=ds)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2cbc1dca788>



Checking For Missing Data :

In [12]:
```
ds.isnull().any()
```

Out[12]:
```
age          False
job          False
marital      False
education    False
default      False
balance      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
deposit      False
dtype: bool
```

Label Encoding :

In [13]:
```
ds.head()
```

Out[13]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
ds['education']=le.fit_transform(ds['education'])
ds['loan']=le.fit_transform(ds['loan'])
ds['deposit']=le.fit_transform(ds['deposit'])
ds['housing']=le.fit_transform(ds['housing'])
ds['default']=le.fit_transform(ds['default'])
ds['month']=le.fit_transform(ds['month'])
ds['job']=le.fit_transform(ds['job'])
ds['poutcome']=le.fit_transform(ds['poutcome'])
ds['contact']=le.fit_transform(ds['contact'])
ds['marital']=le.fit_transform(ds['marital'])
```

In [15]:
```
ds.head()
```

Out[15]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	0	1	1	0	2343	1	0	2	5	8	1042	1	-1	0	3	1
1	56	0	1	1	0	45	0	0	2	5	8	1467	1	-1	0	3	1
2	41	9	1	1	0	1270	1	0	2	5	8	1389	1	-1	0	3	1
3	55	7	1	1	0	2476	1	0	2	5	8	579	1	-1	0	3	1
4	54	0	1	2	0	184	0	0	2	5	8	673	2	-1	0	3	1

One Hot Encoding:

In [16]:
```
x=ds.iloc[:,0:16].values
y=ds.iloc[:,16:17].values
```

In [17]:
```
from sklearn.preprocessing import OneHotEncoder   #only strings values are converted not numerical
one=OneHotEncoder()
p=one.fit_transform(x[:,1:2]).toarray()
q=one.fit_transform(x[:,2:3]).toarray()
r=one.fit_transform(x[:,3:4]).toarray()
s=one.fit_transform(x[:,8:9]).toarray()
t=one.fit_transform(x[:,10:11]).toarray()
v=one.fit_transform(x[:,15:16]).toarray()
x=np.delete(x,[1,2,3,8,10,15],axis=1)
x=np.concatenate((v,t,s,r,q,p,x),axis=1)
```

In [18]:
```
x.shape
```

Out[18]:
```
(11162, 48)
```

Splitting The Dataset Into Train set And Test set:

In [19]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Feature Scaling:

In [20]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

Training And Testing The Model:

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
rrc=RandomForestClassifier(n_estimators=100,criterion='entropy',random_state=0)
rrc.fit(x_train,y_train)
```

```
C:\Users\kuram\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[21]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

In [22]:
```
import pickle
pickle.dump(rrc,open("bank.pkl","wb"))
```

In [23]:
```
y_pred  = rrc.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[23]:
```
array([[1008,  197],
       [ 130,  898]], dtype=int64)
```

In [24]:
```
import sklearn.metrics as metrics
fpr,tpr,threshold=metrics.roc_curve(y_test,y_pred)
roc_auc=metrics.auc(fpr,tpr)
```

Evaluation:

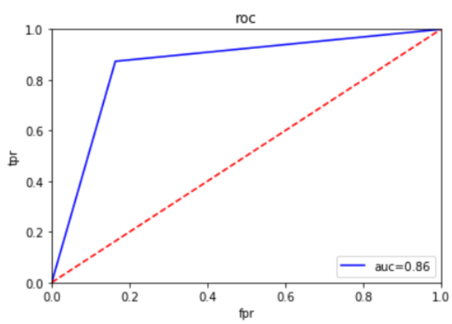
In [25]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[25]:
```
0.8535602328705777
```

In [26]:
```
import matplotlib.pyplot as plt
plt.title("roc")
plt.plot(fpr,tpr,'b',label='auc=%0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('tpr')
plt.xlabel('fpr')
```

Out[26]:
```
Text(0.5, 0, 'fpr')
```

Out[26]:
```
Text(0.5, 0, 'fpr')
```



DONE