

```
#import packages
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split,GridSearchCV,RandomizedSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier
from imblearn.over_sampling import SMOTE,RandomOverSampler,ADASYN
from imblearn.under_sampling import RandomUnderSampler,NearMiss
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report,ConfusionMatrixDisplay
warnings.filterwarnings('ignore')
```

```
#load dataset
df=pd.read_csv("/content/drive/MyDrive/MLDataset/IT_customer_churn.csv")
df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	Female	0	Yes	No	1	No	No
1	Male	0	No	No	34	Yes	No
2	Male	0	No	No	2	Yes	No
3	Male	0	No	No	45	No	No
4	Female	0	No	No	2	Yes	No

```
#analysis of data
df.shape
```

```
(7043, 20)
```

```
df.isna().sum()
```

```
gender      0
SeniorCitizen  0
Partner      0
Dependents   0
tenure       0
PhoneService  0
MultipleLines  0
InternetService  0
OnlineSecurity  0
OnlineBackup  0
DeviceProtection  0
TechSupport   0
StreamingTV    0
```

```

StreamingMovies    0
Contract           0
PaperlessBilling   0
PaymentMethod      0
MonthlyCharges     0
TotalCharges       0
Churn              0
dtype: int64

```

df.dtypes

```

gender            object
SeniorCitizen     int64
Partner           object
Dependents        object
tenure            int64
PhoneService      object
MultipleLines     object
InternetService   object
OnlineSecurity    object
OnlineBackup      object
DeviceProtection  object
TechSupport       object
StreamingTV       object
StreamingMovies   object
Contract          object
PaperlessBilling  object
PaymentMethod     object
MonthlyCharges    float64
TotalCharges      object
Churn             object
dtype: object

```

df["TotalCharges"].values

```

array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
      dtype=object)

```

#converting to numeric and then check missing values

```

df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()

```

```

gender            0
SeniorCitizen     0
Partner           0
Dependents        0
tenure            0
PhoneService      0
MultipleLines     0
InternetService   0
OnlineSecurity    0
OnlineBackup      0
DeviceProtection  0
TechSupport       0
StreamingTV       0
StreamingMovies   0
Contract          0
PaperlessBilling  0
PaymentMethod     0
MonthlyCharges    0
TotalCharges      11
Churn             0
dtype: int64

```

df.dtypes

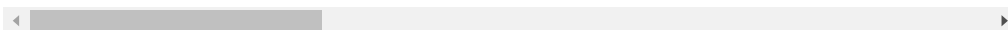
```

gender          object
SeniorCitizen   int64
Partner         object
Dependents      object
tenure          int64
PhoneService    object
MultipleLines   object
InternetService object
OnlineSecurity  object
OnlineBackup    object
DeviceProtection object
TechSupport     object
StreamingTV     object
StreamingMovies object
Contract        object
PaperlessBilling object
PaymentMethod   object
MonthlyCharges  float64
TotalCharges    float64
Churn           object
dtype: object

```

```
df.loc[df['TotalCharges'].isnull() == True]
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
<b>488</b>	Female	0	Yes	Yes	0	No	No ph ser
<b>753</b>	Male	0	No	Yes	0	Yes	
<b>936</b>	Female	0	Yes	Yes	0	Yes	
<b>1082</b>	Male	0	Yes	Yes	0	Yes	
<b>1340</b>	Female	0	Yes	Yes	0	No	No ph ser
<b>3331</b>	Male	0	Yes	Yes	0	Yes	
<b>3826</b>	Male	0	Yes	Yes	0	Yes	
<b>4380</b>	Female	0	Yes	Yes	0	Yes	
<b>5218</b>	Male	0	Yes	Yes	0	Yes	
<b>6670</b>	Female	0	Yes	Yes	0	Yes	
<b>6754</b>	Male	0	No	Yes	0	Yes	



```

#dropping the row contain Null value
df.dropna(how = 'any', inplace = True)
df.shape

```

```
(7032, 20)
```

```

#checking unique values of each column
def printunique(df):
    for i in df:

```

```

if df[i].dtypes=='object':
    print(f'{i}: {df[i].unique()}')
printunique(df)

gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
'Credit card (automatic)']
Churn: ['No' 'Yes']

df.replace('No internet service','No',inplace=True)
df.replace('No phone service','No',inplace=True)

```

```

printunique(df)

gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes']
OnlineBackup: ['Yes' 'No']
DeviceProtection: ['No' 'Yes']
TechSupport: ['No' 'Yes']
StreamingTV: ['No' 'Yes']
StreamingMovies: ['No' 'Yes']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
'Credit card (automatic)']
Churn: ['No' 'Yes']

sns.countplot(x="Churn",data=df)

```

<Axes: xlabel='Churn', ylabel='count'>



```
100*df['Churn'].value_counts()/len(df['Churn'])
```

```
No    73.421502
```

```
Yes    26.578498
```

```
Name: Churn, dtype: float64
```



#Data is highly imbalanced, ratio = 73:27

```
df['Churn'].value_counts()
```

```
No    5163
```

```
Yes    1869
```

```
Name: Churn, dtype: int64
```



```
t_c_n = df[df.Churn=='No'].tenure
```

```
t_c_y = df[df.Churn=='Yes'].tenure
```

```
plt.xlabel("tenure")
```

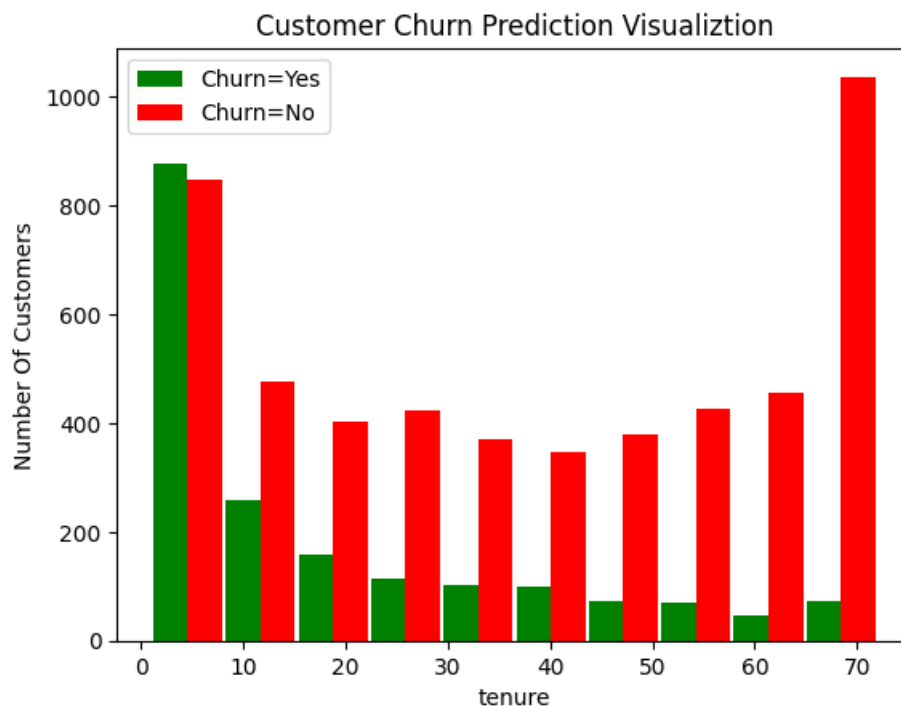
```
plt.ylabel("Number Of Customers")
```

```
plt.title("Customer Churn Prediction Visualiztion")
```

```
plt.hist([t_c_y, t_c_n], rwidth=0.95, color=['green', 'red'], label=['Churn=Yes', 'Churn=No'])
```

```
plt.legend()
```

```
plt.show()
```



```
m_c_n = df[df.Churn=='No'].MonthlyCharges
```

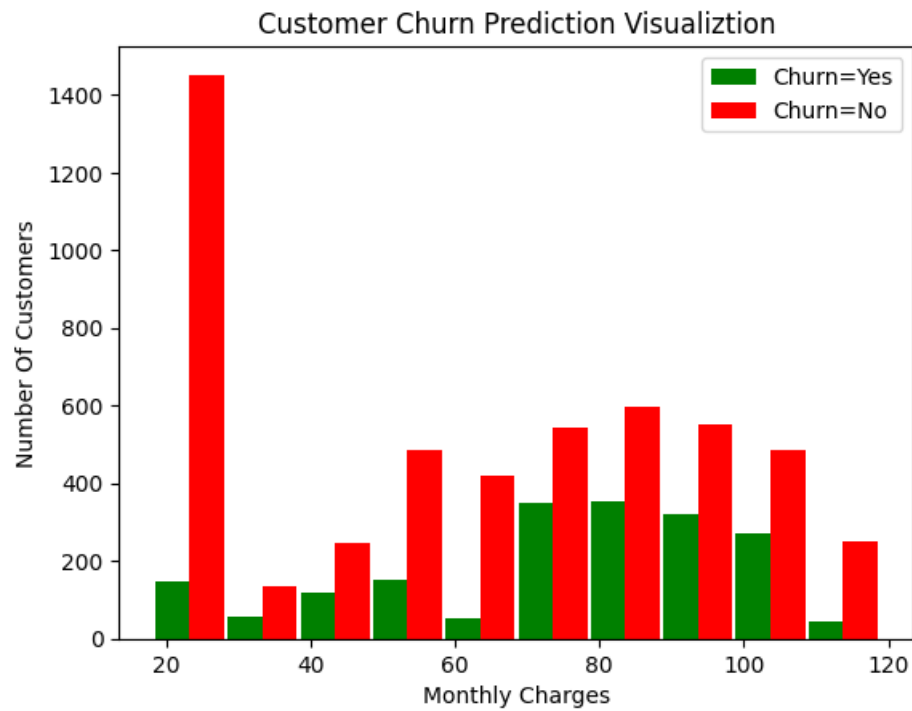
```
m_c_y = df[df.Churn=='Yes'].MonthlyCharges
```

```
plt.xlabel("Monthly Charges")
```

```
plt.ylabel("Number Of Customers")
```

```
plt.title("Customer Churn Prediction Visualiztion")
```

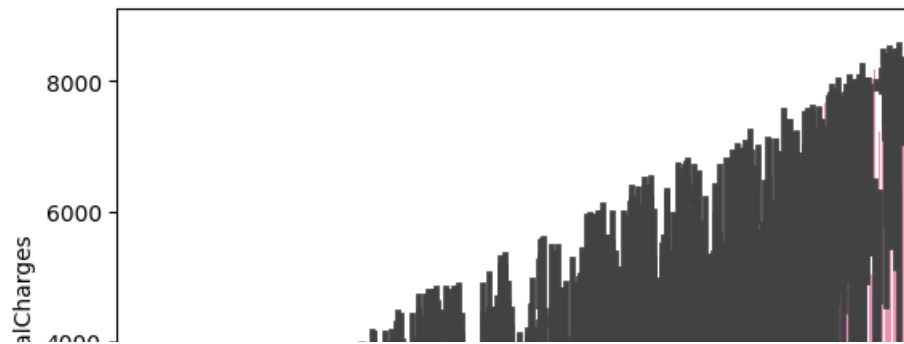
```
plt.hist([m_c_y, m_c_n], rwidth=0.95, color=['green', 'red'], label=['Churn=Yes', 'Churn=No'])  
plt.legend()  
plt.show()
```



```
for i, predictor in enumerate(df.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):  
    plt.figure(i)  
    sns.countplot(data=df, x=predictor, hue='Churn')
```

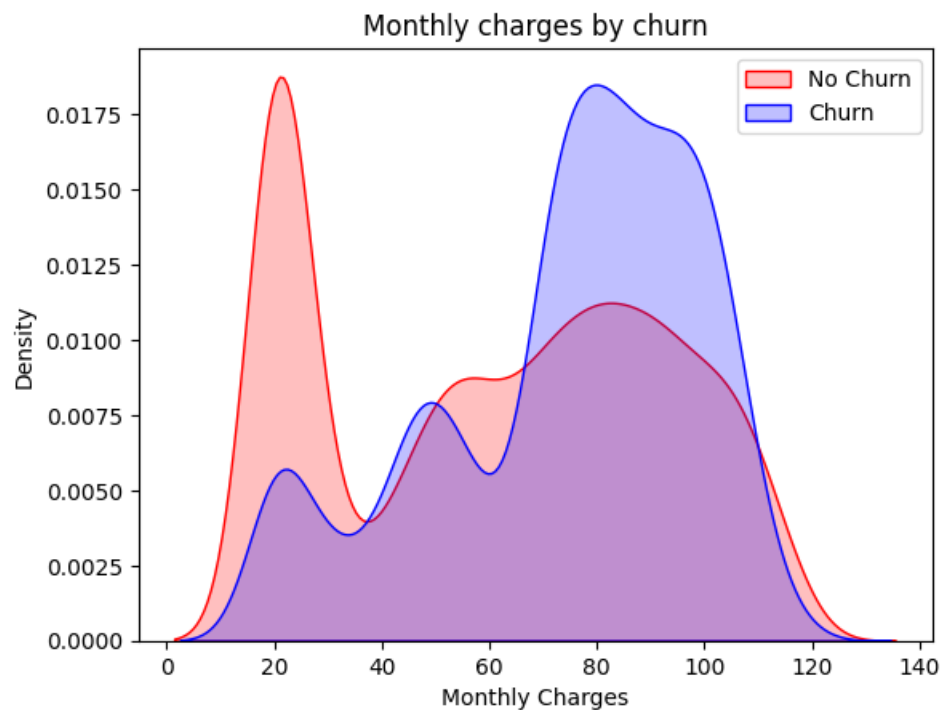


<Axes: xlabel='MonthlyCharges', ylabel='TotalCharges'>



```
mon = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 0) ],
                  color="Red", shade = True)
mon = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 1) ],
                  ax =mon, color="Blue", shade= True)
mon.legend(["No Churn","Churn"],loc='upper right')
mon.set_ylabel('Density')
mon.set_xlabel('Monthly Charges')
mon.set_title('Monthly charges by churn')
```

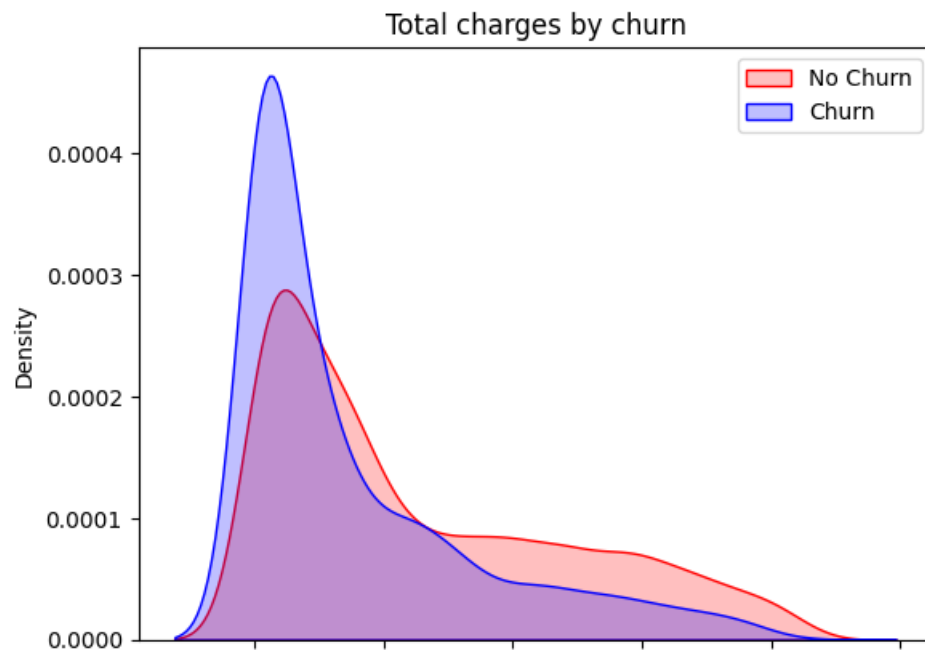
Text(0.5, 1.0, 'Monthly charges by churn')



123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100101102103104105106107108109110111112113114115116117118119120121122123124125126127128129130131132133134135136137138139140141142143144145146147148149150151152153154155156157158159160161162163164165166167168169170171172173174175176177178179180181182183184185186187188189190191192193194195196197198199200201202203204205206207208209210211212213214215216217218219220221222223224225226227228229230231232233234235236237238239240241242243244245246247248249250251252253254255256257258259260261262263264265266267268269270271272273274275276277278279280281282283284285286287288289290291292293294295296297298299300301302303304305306307308309310311312313314315316317318319320321322323324325326327328329330331332333334335336337338339340341342343344345346347348349350351352353354355356357358359360361362363364365366367368369370371372373374375376377378379380381382383384385386387388389390391392393394395396397398399400401402403404405406407408409410411412413414415416417418419420421422423424425426427428429430431432433434435436437438439440441442443444445446447448449450451452453454455456457458459460461462463464465466467468469470471472473474475476477478479480481482483484485486487488489490491492493494495496497498499500501502503504505506507508509510511512513514515516517518519520521522523524525526527528529530531532533534535536537538539540541542543544545546547548549550551552553554555556557558559560561562563564565566567568569570571572573574575576577578579580581582583584585586587588589590591592593594595596597598599600601602603604605606607608609610611612613614615616617618619620621622623624625626627628629630631632633634635636637638639640641642643644645646647648649650651652653654655656657658659660661662663664665666667668669670671672673674675676677678679680681682683684685686687688689690691692693694695696697698699700701702703704705706707708709710711712713714715716717718719720721722723724725726727728729730731732733734735736737738739740741742743744745746747748749750751752753754755756757758759760761762763764765766767768769770771772773774775776777778779780781782783784785786787788789790791792793794795796797798799800801802803804805806807808809810811812813814815816817818819820821822823824825826827828829830831832833834835836837838839840841842843844845846847848849850851852853854855856857858859860861862863864865866867868869870871872873874875876877878879880881882883884885886887888889890891892893894895896897898899900901902903904905906907908909910911912913914915916917918919920921922923924925926927928929930931932933934935936937938939940941942943944945946947948949950951952953954955956957958959960961962963964965966967968969970971972973974975976977978979980981982983984985986987988989990991992993994995996997998999100010011002100310041005100610071008100910101011101210131014101510161017101810191020102110221023102410251026102710281029103010311032103310341035103610371038103910401041104210431044104510461047104810491050105110521053105410551056105710581059106010611062106310641065106610671068106910701071107210731074107510761077107810791080108110821083108410851086108710881089109010911092109310941095109610971098109911001101110211031104110511061107110811091110111111121113111411151116111711181119112011211122112311241125112611271128112911301131113211331134113511361137113811391140114111421143114411451146114711481149115011511152115311541155115611571158115911601161116211631164116511661167116811691170117111721173117411751176117711781179118011811182118311841185118611871188118911901191119211931194119511961197119811991200120112021203120412051206120712081209121012111212121312141215121612171218121912201221122212231224122512261227122812291230123112321233123412351236123712381239124012411242124312441245124612471248124912501251125212531254125512561257125812591260126112621263126412651266126712681269127012711272127312741275127612771278127912801281128212831284128512861287128812891290129112921293129412951296129712981299130013011302130313041305130613071308130913101311131213131314131513161317131813191320132113221323132413251326132713281329133013311332133313341335133613371338133913401341134213431344134513461347134813491350135113521353135413551356135713581359136013611362136313641365136613671368136913701371137213731374137513761377137813791380138113821383138413851386138713881389139013911392139313941395139613971398139914001401140214031404140514061407140814091410141114121413141414151416141714181419142014211422142314241425142614271428142914301431143214331434143514361437143814391440144114421443144414451446144714481449145014511452145314541455145614571458145914601461146214631464146514661467146814691470147114721473147414751476147714781479148014811482148314841485148614871488148914901491149214931494149514961497149814991500150115021503150415051506150715081509151015111512151315141515151615171518151915201521152215231524152515261527152815291530153115321533153415351536153715381539154015411542154315441545154615471548154915501551155215531554155515561557155815591560156115621563156415651566156715681569157015711572157315741575157615771578157915801581158215831584158515861587158815891590159115921593159415951596159715981599160016011602160316041605160616071608160916101611161216131614161516161617161816191620162116221623162416251626162716281629163016311632163316341635163616371638163916401641164216431644164516461647164816491650165116521653165416551656165716581659166016611662166316641665166616671668166916701671167216731674167516761677167816791680168116821683168416851686168716881689169016911692169316941695169616971698169917001701170217031704170517061707170817091710171117121713171417151716171717181719172017211722172317241725172617271728172917301731173217331734173517361737173817391740174117421743174417451746174717481749175017511752175317541755175617571758175917601761176217631764176517661767176817691770177117721773177417751776177717781779178017811782178317841785178617871788178917901791179217931794179517961797179817991800180118021803180418051806180718081809181018111812181318141815181618171818181918201821182218231824182518261827182818291830183118321833183418351836183718381839184018411842184318441845184618471848184918501851185218531854185518561857185818591860186118621863186418651866186718681869187018711872187318741875187618771878187918801881188218831884188518861887188818891890189118921893189418951896189718981899190019011902190319041905190619071908190919101911191219131914191519161917191819191920192119221923192419251926192719281929193019311932193319341935193619371938193919401941194219431944194519461947194819491950195119521953195419551956195719581959196019611962196319641965196619671968196919701971197219731974197519761977197819791980198119821983198419851986198719881989199019911992199319941995199619971998199920002001200220032004200520062007200820092010201120122013201420152016201720182019202020212022202320242025202620272028202920302031203220332034203520362037203820392040204120422043204420452046204720482049205020512052205320542055205620572058205920602061206220632064206520662067206820692070207120722073207420752076207720782079208020812082208320842085208620872088208920902091209220932094209520962097209820992100210121022103210421052106210721082109211021112112211321142115211621172118211921202121212221232124212521262127212821292130213121322133213421352136213721382139214021412142214321442145214621472148214921502151215221532154215521562157215821592160216121622163216421652166216721682169217021712172217321742175217621772178217921802181218221832184218521862187218821892190219121922193219421952196219721982199220022012202220322042205220622072208220922102211221222132214221522162217221822192220222122222223222422252226222722282229223022312232223322342235223622372238223922402241224222432244224522462247224822492250225122522253225422552256225722582259226022612262226322642265226622672268226922702271227222732274227522762277227822792280228122822283228422852286228722882289229022912292229322942295229622972298229923002301230223032304230523062307230823092310231123122313231423152316231723182319232023212322232323242325232623272328232923302331233223332334233523362337233823392340234123422343234423452346234723482349235023512352235323542355235623572358235923602361236223632364236523662367236823692370237123722373237423752376237723782379238023812382238323842385238623872388238923902391239223932394239523962397239823992400240124022403240424052406240724082409241024112412241324142415241624172418241924202421242224232424242524262427242824292430243124322433243424352436243724382439244024412442244324442445244624472448244924502451245224532454245524562457245824592460246124622463246424652466246724682469247024712472247324742475247624772478247924802481248224832484248524862487248824892490249124922493249424952496249724982499250025012502250325042505250625072508250925102511251225132514251525162517251825192520252125222523252425252526252725282529253025312532253325342535253625372538253925402541254225432544254525462547254825492550255125522553255425552556255725582559256025612562256325642565256625672568256925702571257225732574257525762577257825792580258125822583258425852586258725882589259025912592259325942595259625972598259926002601260226032604260526062607260826092610261126122613261426152616261726182619262026212622262326242625262626272628262926302631263226332634263526362637263826392640264126422643264426452646264726482649265026512652265326542655265626572658265926602661266226632664266526662667266826692670267126722673267426752676267726782679268026812682268326842685268626872688268926902691269226932694269526962697269826992700270127022703270427052706270727082709271027112712271327142715271627172718271927202721272227232724272527262727272827292730273127322733273427352736273727382739274027412742274327442745274627472748274927502751275227532754275527562757275827592760276127622763276427652766276727682769277027712772277327742775277627772778277927802781278227832784278527862787278827892790279127922793279427952796279727982799280028012802280328042805280628072808280928102811281228132814281528162817281828192820282128222823282428252826282728282829283028312832283328342835283628372838283928402841284228432844284528462847284828492850285128522853285428552856285728582859286028612862286328642865286628672868286928702871287228732874287528762877287828792880288128822883288428852886288728882889289028912892289328942895289628972898289929002901290229032904290529062907290829092910291129122913291429152916291729182919292029212922292329242925292629272928292929302931293229332934293529362937293829392940294129422943294429452946294729482949295029512952295329542955295629572958295929602961296229632964296529662967296829692970297129722973297429752976297729782979298029812982298329842985298629872988298929902991299229932994299529962997299829993000300130023003300430053006300730083009301030113012301330143015301630173018301930203021302230233024302530263027302830293030303130323033303430353036303730383039304030413042304330443045304630473048304930503051305230533054305530563057305830593060306130623063306430653066306730683069307030713072307330743075307630773078307930803081308230833084308530863087308830893090309130923093309430953096309730983099310031013102310331043105310631073108310931103111311231133114311531163117311831193120312131223123312431253126312731283129313031313132313331343135313631373138313931403141314231433144314531463147314831493150315131523153315431553156315731583159316031613162316331643165316631673168316931703171317231733174317531763177317831793180318131823183318431853186318731883189319031913192319331943195319631973198319932003201320232033204320532063207320832093210321132123213321432153216321732183219322032213



```
Text(0.5, 1.0, 'Total charges by churn')
```



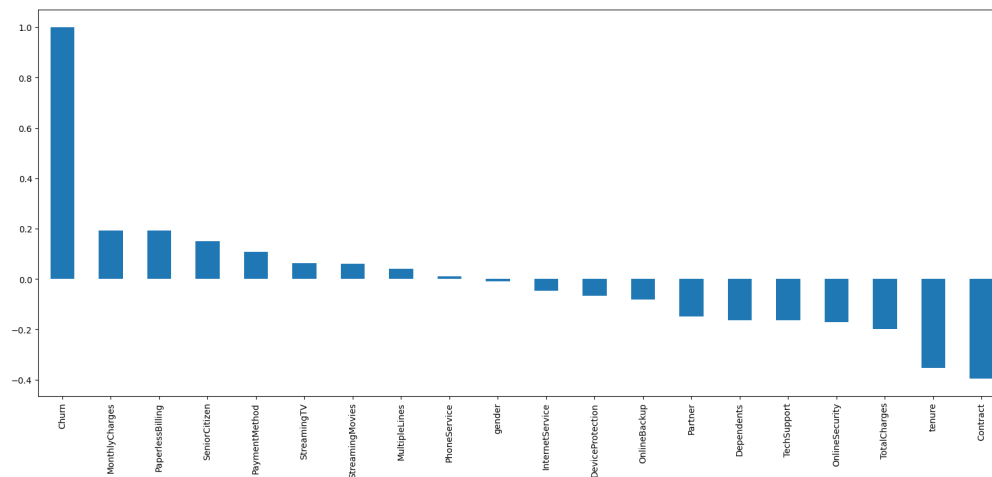
```
#as higher Churn at lower Total Charges
```



```
plt.figure(figsize=(20,8))
```

```
df.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

<Axes: >



#HIGH Churn seen in case of Monthly charges,paperless billing etc..

#LOW Churn is seen in case of contract,tenure,totalcharges etc..

#Factors like gender, Availability of PhoneService and # of multiple lines have almost NO impact on Churn

1000 |  |

Double-click (or enter) to edit


1000 |  |

```
X=df.iloc[:, :-1]
```

```
X
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
0	0	0	1	0	1	0	
1	1	0	0	0	34	1	
2	1	0	0	0	2	1	
3	1	0	0	0	45	0	
4	0	0	0	0	2	1	
...	...	...	...	...	...	...	...
7038	1	0	1	1	24	1	
7039	0	0	1	1	72	1	
7040	0	0	1	1	11	0	
7041	1	1	1	0	4	1	
7042	1	0	0	0	66	1	

7032 rows × 19 columns

◀  ▶

|  | Churn ||

```
y=df.iloc[:, -1]
```

```
y
```

```
0      0
1      0
2      1
3      0
4      1
..
7038   0
7039   0
7040   0
7041   1
7042   0
```

Name: Churn, Length: 7032, dtype: int64

500 |  |

```
#ms=MinMaxScaler()
```


```
#X_sc=ms.fit_transform(X)
```

```
#X_sc
```

DeviceProtection

```
st=StandardScaler()
```

```
X_st=st.fit_transform(X)
```

3000 |  |  100% ||

```
sm=SMOTE(random_state=10)
```

```
X_sm,y_sm=sm.fit_resample(X_st,y)
```

```
y_sm.value_counts()
```

```
0    5163
1    5163
Name: Churn, dtype: int64
```

```
X_train,X_test,y_train,y_test=train_test_split(X_sm,y_sm,random_state=1,test_size=0.02)
```

```
#pca=PCA(n_components=1,random_state=1)
#X_train=pca.fit_transform(X_train)
#X_test=pca.transform(X_test)
#X_train
```

```
recnsupport
```

```
knn=KNeighborsClassifier()
params={'n_neighbors':[3,5,7,9], 'weights':['uniform','distance'], 'algorithm':['auto','ball_tree','kd_tree','brute']}
clf=GridSearchCV(knn,params,cv=10,scoring='accuracy')
clf.fit(X_train,y_train)
print(clf.best_params_)
```

```
{'algorithm': 'ball_tree', 'n_neighbors': 3, 'weights': 'distance'}
```

```
knn=KNeighborsClassifier(algorithm='auto',n_neighbors=3,weights='distance')
svm=SVC(gamma="auto",kernel="rbf")
nb=GaussianNB()
rf=RandomForestClassifier(criterion='entropy', n_estimators=100,max_depth=8,max_features=6,min_samples_leaf=7)
gb=GradientBoostingClassifier(n_estimators=100)
dc=DecisionTreeClassifier(criterion='entropy',max_depth=6, min_samples_leaf=8,random_state=5)
lg=LogisticRegression()
```

```
lst2=[knn,svm,nb,rf,gb,dc,lg]
for i in lst2:
    print(i)
    i.fit(X_train,y_train)
    y_pred=i.predict(X_test)
    print(classification_report(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=3, weights='distance')
precision    recall  f1-score   support
```

```
0      0.88      0.73      0.80      102
1      0.77      0.90      0.83      105
```

```
accuracy          0.82      207
macro avg         0.83      0.82      0.81      207
weighted avg      0.83      0.82      0.81      207
```

```
SVC(gamma='auto')
precision    recall  f1-score   support
```

```
0      0.84      0.79      0.82      102
1      0.81      0.86      0.83      105
```

```
accuracy          0.83      207
macro avg         0.83      0.83      0.83      207
weighted avg      0.83      0.83      0.83      207
```

```
GaussianNB()
precision    recall  f1-score   support
```

```
0      0.80      0.65      0.72      102
1      0.71      0.85      0.77      105
```

accuracy			0.75	207
macro avg	0.76	0.75	0.75	207
weighted avg	0.76	0.75	0.75	207

```
RandomForestClassifier(criterion='entropy', max_depth=8, max_features=6,
                        min_samples_leaf=7)
```

	precision	recall	f1-score	support
0	0.88	0.77	0.82	102
1	0.80	0.90	0.85	105

accuracy			0.84	207
macro avg	0.84	0.83	0.83	207
weighted avg	0.84	0.84	0.84	207

```
GradientBoostingClassifier()
```

	precision	recall	f1-score	support
0	0.89	0.83	0.86	102
1	0.85	0.90	0.87	105

accuracy			0.86	207
macro avg	0.87	0.86	0.86	207
weighted avg	0.87	0.86	0.86	207

```
DecisionTreeClassifier(criterion='entropy', max_depth=6, min_samples_leaf=8,
                       random_state=5)
```

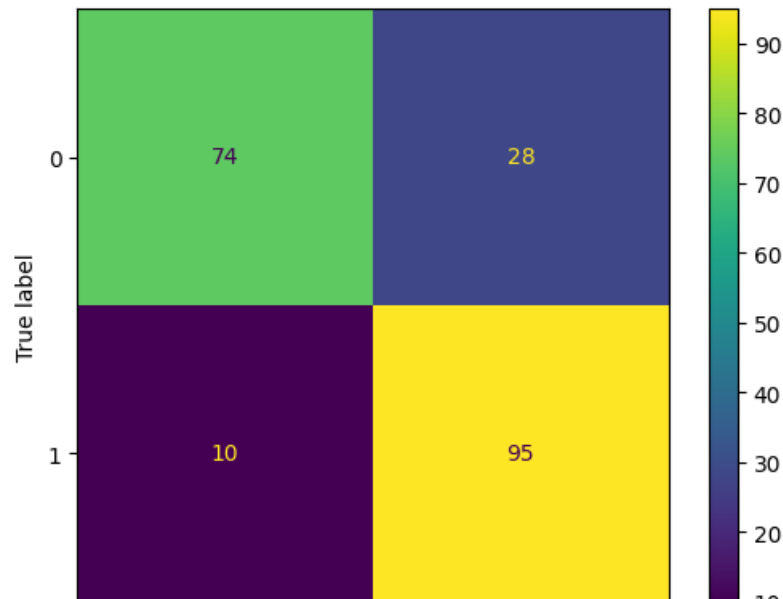
	precision	recall	f1-score	support
0	0.84	0.83	0.84	102
1	0.84	0.85	0.84	105

1500

```
lst2=[knn,svm]
for i in lst2:
    print(i)
    i.fit(X_train,y_train)
    y_pred=i.predict(X_test)

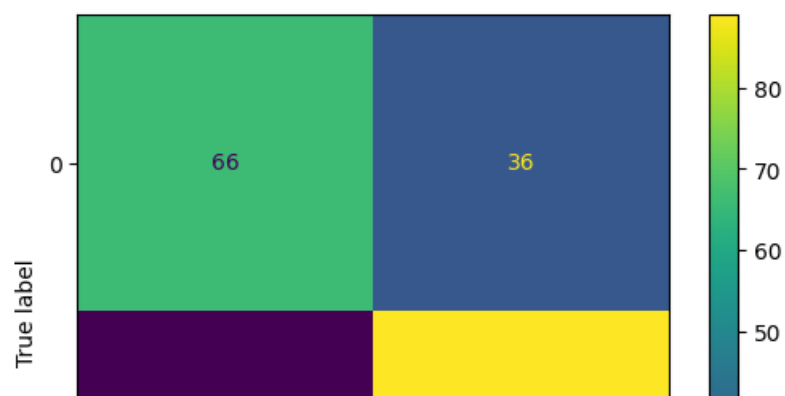
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=3, weights='distance')  
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f232a6  
SVC(gamma='auto')  
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f236a1
```

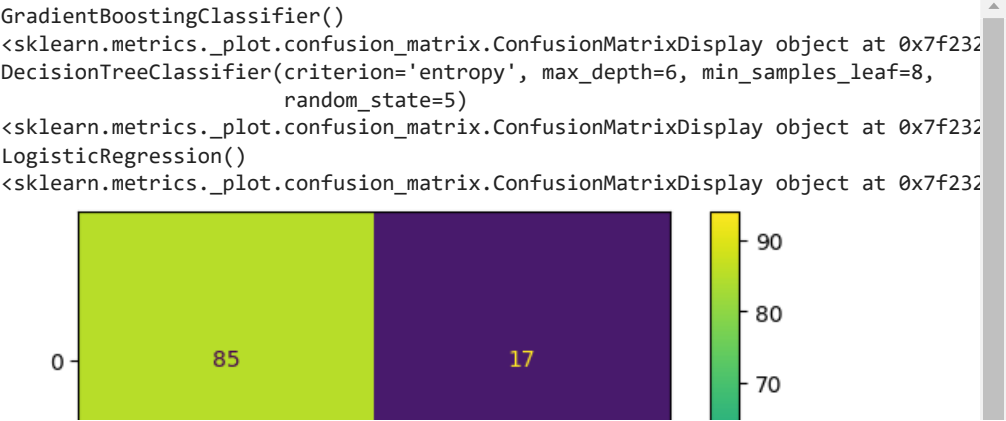


```
lst2=[nb,rf]  
for i in lst2:  
    print(i)  
    i.fit(X_train,y_train)  
    y_pred=i.predict(X_test)  
    print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
GaussianNB()  
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f232a4  
RandomForestClassifier(criterion='entropy', max_depth=8, max_features=6,  
                        min_samples_leaf=7)  
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f232a6
```



```
lst2=[gb,dc,lg]  
for i in lst2:  
    print(i)  
    i.fit(X_train,y_train)  
    y_pred=i.predict(X_test)  
    print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```



```
d={'Model':['K-Nearest Neighbors','Support Vector Classifier','GaussianNB','Random Forest Classifier','GradientBoosti  
df_new=pd.DataFrame(d,index=[1,2,3,4,5,6,7],columns=['Model','Prediction_Accuracy'])  
df_new.style.highlight_max(subset=['Prediction_Accuracy'],color='green')
```

	Model	Prediction_Accuracy
1	K-Nearest Neighbors	0.820000
2	Support Vector Classifier	0.830000
3	GaussianNB	0.750000
4	Random Forest Classifier	0.850000
5	GradientBoostingClassifier	0.860000
6	DecisionTreeClassifier	0.840000
7	logisticRegression	0.800000

A horizontal bar chart showing the prediction accuracy for each model. The bars are colored green for the highest accuracy (GradientBoostingClassifier) and purple for the lowest (logisticRegression).

```
df_new.style.highlight_min(subset=['Prediction_Accuracy'],color='red')
```

	Model	Prediction_Accuracy
1	K-Nearest Neighbors	0.820000
2	Support Vector Classifier	0.830000
3	GaussianNB	0.750000
4	Random Forest Classifier	0.850000
5	GradientBoostingClassifier	0.860000
6	DecisionTreeClassifier	0.840000
7	logisticRegression	0.800000

A horizontal bar chart showing the prediction accuracy for each model. The bars are colored purple for the lowest accuracy (logisticRegression) and green for the highest (GradientBoostingClassifier).

```
y=[82,83,75,85,86,84,80]  
x=['Knn','Svm','GNB','RFC','Gradient','DTree','Logistic']  
plt.xlabel("Models")  
plt.ylabel("Accuracy")  
plt.title("Models VS Accuracy")  
plt.bar(x,y,color="b",width=0.4)  
plt.show()
```

