In [1]:
```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
print(__version__)
import cufflinks as cf
init_notebook_mode(connected=True)
cf.go_offline()
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd
```

```
5.6.0
```

In [2]:
```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [86]:
```
df=pd.read_csv("C://Users//abhij//Desktop//data mining//Loan Prediction Data.csv")
df.head()
```

Out[86]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappli
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

In [87]:
```
df.shape
```

Out[87]:
```
(981, 13)
```

In [88]:
```
df.size
```

Out[88]:
```
12753
```

In [89]:
```
df.describe()
```

Out[89]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	981.000000	981.000000	954.000000	961.000000	902.000000
mean	5179.795107	1601.916330	142.511530	342.201873	0.835920
std	5695.104533	2718.772806	77.421743	65.100602	0.370553
min	0.000000	0.000000	9.000000	6.000000	0.000000
25%	2875.000000	0.000000	100.000000	360.000000	1.000000
50%	3800.000000	1110.000000	126.000000	360.000000	1.000000
75%	5516.000000	2365.000000	162.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

In [90]:
```
#transpose
df.describe().T
```

Out[90]:

	count	mean	std	min	25%	50%	75%	max
ApplicantIncome	981.0	5179.795107	5695.104533	0.0	2875.0	3800.0	5516.0	81000.0
CoapplicantIncome	981.0	1601.916330	2718.772806	0.0	0.0	1110.0	2365.0	41667.0
LoanAmount	954.0	142.511530	77.421743	9.0	100.0	126.0	162.0	700.0
Loan_Amount_Term	961.0	342.201873	65.100602	6.0	360.0	360.0	360.0	480.0
Credit_History	902.0	0.835920	0.370553	0.0	1.0	1.0	1.0	1.0

In [91]:
```
#Filling Null Values
df.isnull().sum()
```

Out[91]:
```
Loan_ID               0
Gender               24
Married               3
Dependents           25
Education             0
Self_Employed        55
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           27
Loan_Amount_Term     20
Credit_History       79
Property_Area         0
Loan_Status         367
dtype: int64
```

In [92]:
```
#Forward fill(Fills NA Values with values beihind in the row)
df.fillna(method='ffill',inplace=True)
```

In [93]:
```
#Backward Fill
df.fillna(method='bfill',inplace=True)
```

In [101…
```
#Checking whether Null Values are left or not
df.isnull().sum()
```

Out[101]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [102…
```
plt.figure(figsize=(10,5))
```

Out[102]:
```
<Figure size 720x360 with 0 Axes>
```

<Figure size 720x360 with 0 Axes>

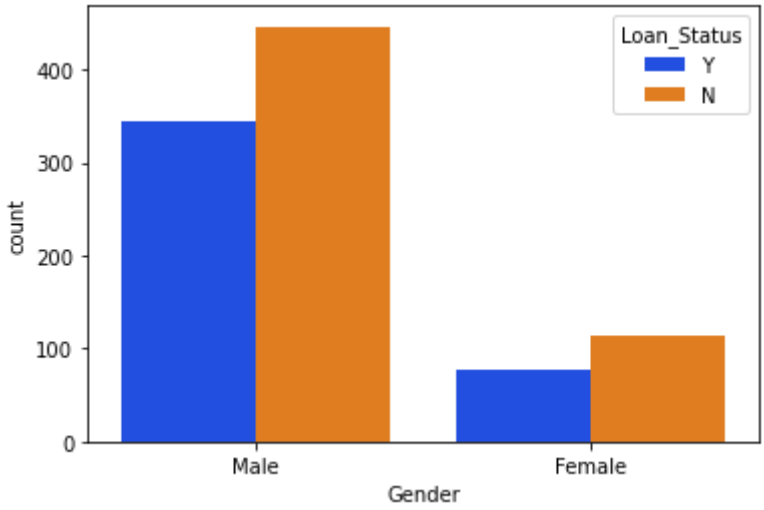
In [106…
```
df.head()
```

Out[106]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappli
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

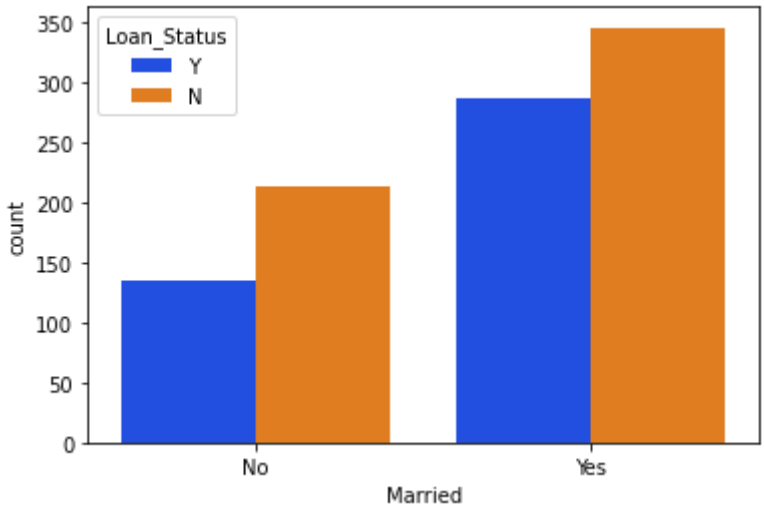
In [107…
```
sns.countplot(data=df,x='Gender',hue='Loan_Status',palette='bright')
```

Out[107]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



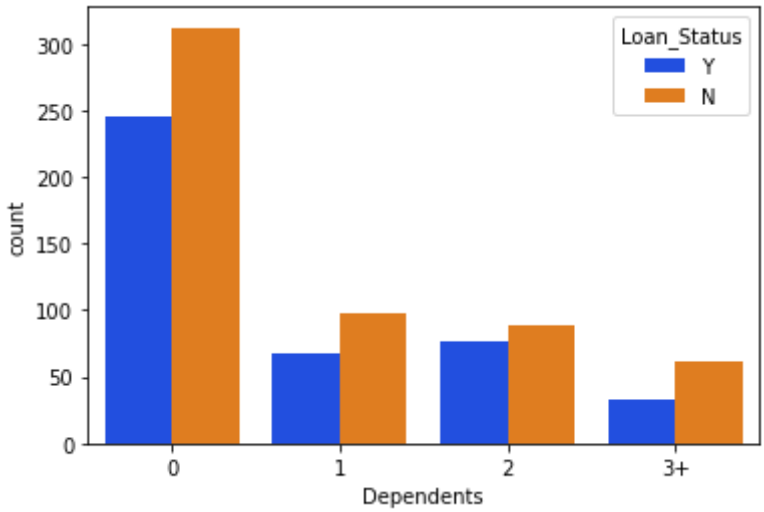
In [110…
```
sns.countplot(data=df,x='Married',hue='Loan_Status',palette='bright')
```

Out[110]:
```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



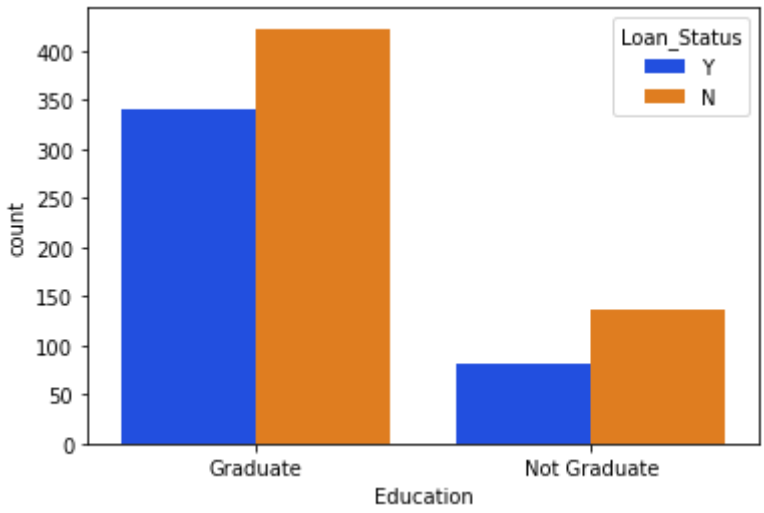
In [111…
```
sns.countplot(data=df,x='Dependents',hue='Loan_Status',palette='bright')
```

Out[111]:
```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



In [112…
```
sns.countplot(data=df,x='Education',hue='Loan_Status',palette='bright')
```

Out[112]:
```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



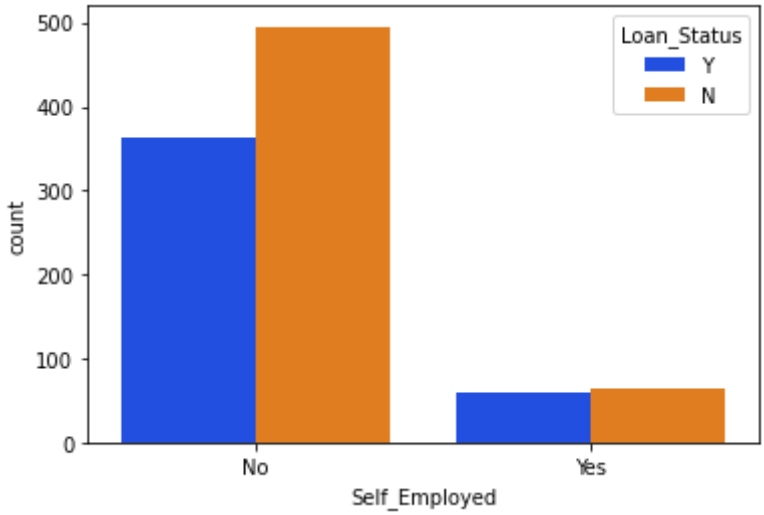
In [113…
```
df.head(2)
```

Out[113]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappli
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	

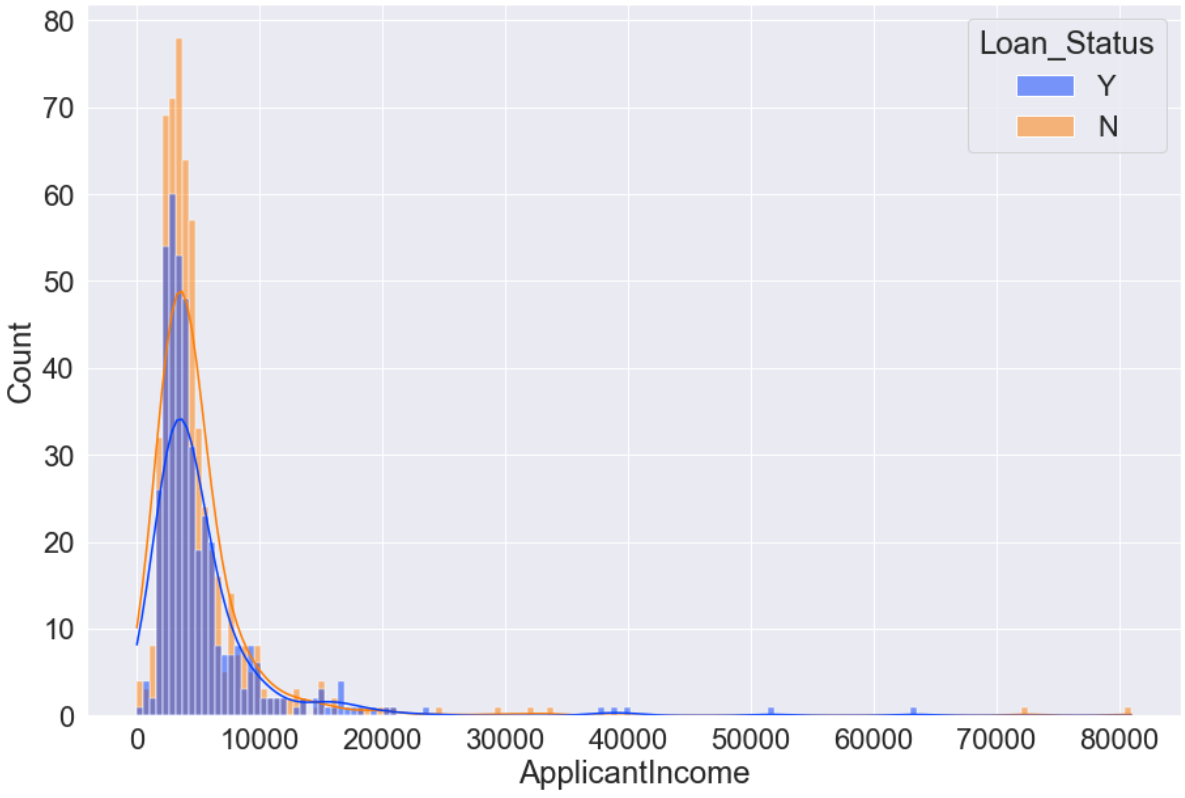
In [120…
```
sns.countplot(data=df,x='Self_Employed',hue='Loan_Status',palette='bright')
```

Out[120]:
```
<AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```

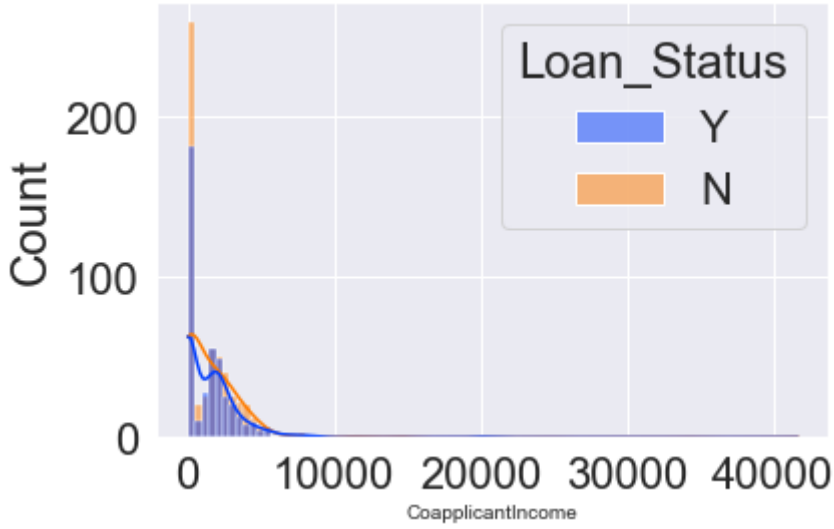


In [138…
```
sns.set(font_scale = 2)
plt.figure(figsize=(15,10))
sns.histplot(data=df,x='ApplicantIncome',hue='Loan_Status',palette='bright',kde=Tru
```

Out[138]:
```
<AxesSubplot:xlabel='ApplicantIncome', ylabel='Count'>
```



In [163…
```
sns.set(font_scale = 2)
sns.histplot(data=df,x='CoapplicantIncome',hue='Loan_Status',palette='bright',kde=
plt.xlabel('CoapplicantIncome', fontsize=10);
```

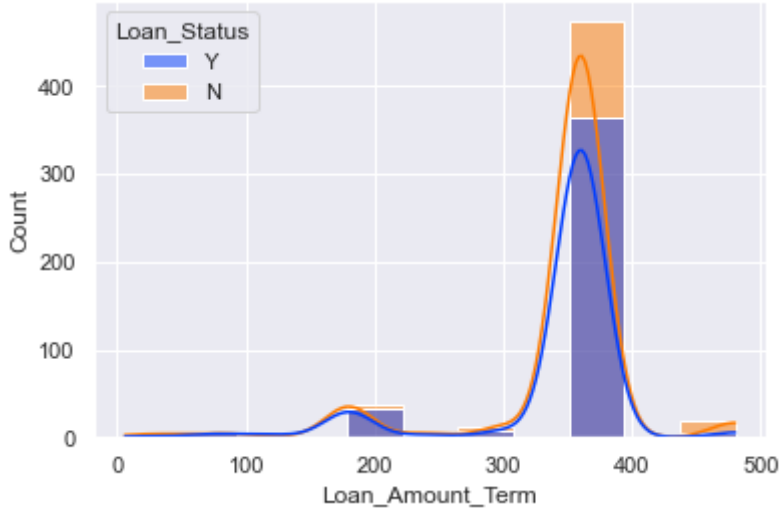


In [161…
```
sns.histplot(data=df,x='LoanAmount',hue='Loan_Status',palette='bright',kde=True)
```

Out[161]:
```
<AxesSubplot:xlabel='LoanAmount', ylabel='Count'>
```

In [160…
```
sns.set(font_scale = 1)
sns.histplot(data=df,x='Loan_Amount_Term',hue='Loan_Status',palette='bright',kde=Tr
```

Out[160]:
```
<AxesSubplot:xlabel='Loan_Amount_Term', ylabel='Count'>
```

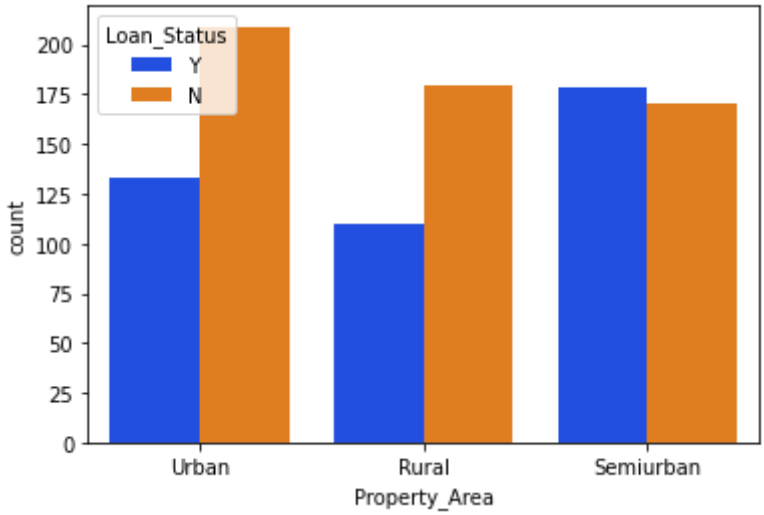


In [115…
```
sns.countplot(data=df,x='Credit_History',hue='Loan_Status',palette='bright')
```

Out[115]: <AxesSubplot:xlabel='Credit_History', ylabel='count'>

In [116…
```
sns.countplot(data=df,x='Property_Area',hue='Loan_Status',palette='bright')
```

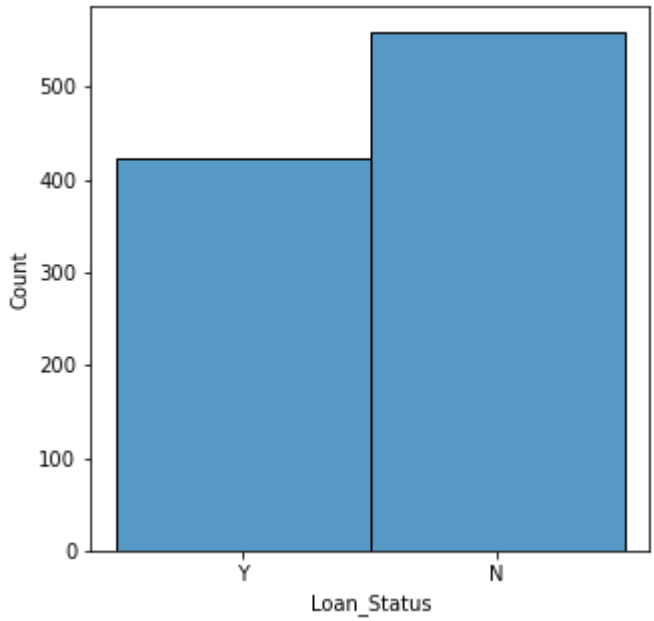
Out[116]: <AxesSubplot:xlabel='Property_Area', ylabel='count'>



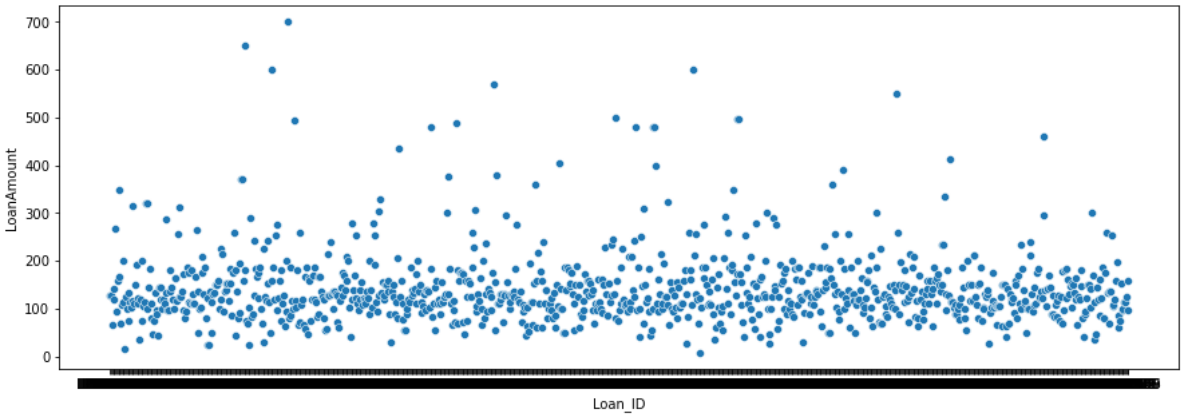
In [ ]:

In [ ]:

In [15]:
```
plt.figure(figsize=(5,5))
sns.histplot(x='Loan_Status',data=df,palette='deep')
plt.show()
```

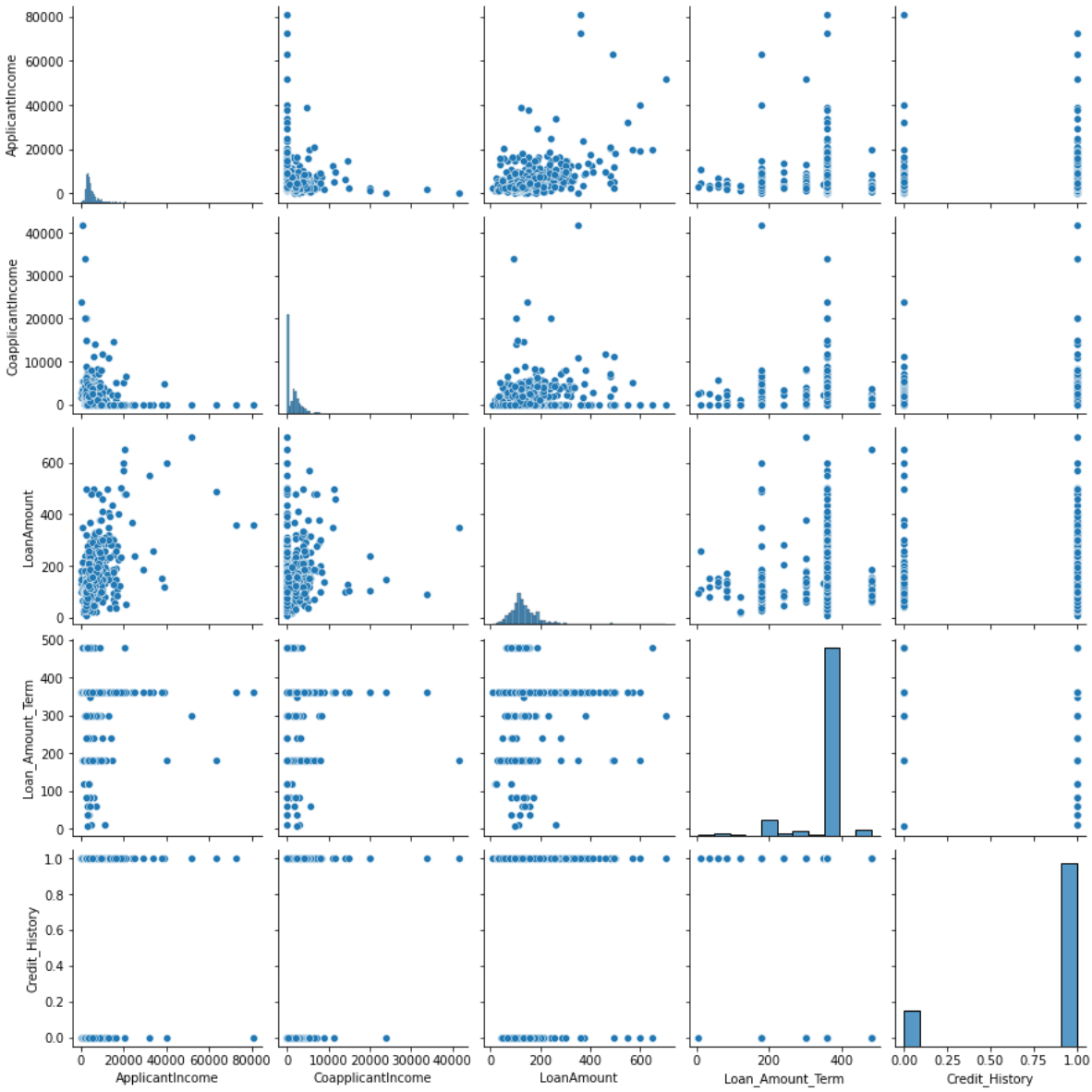


In [16]:
```
plt.figure(figsize=(15,5))
sns.scatterplot(x="Loan_ID",y="LoanAmount",data=df,palette='bright')
plt.show()
```



In [17]:
```
plt.figure(figsize=(10,5))
sns.pairplot(df)
plt.show()
```

```
<Figure size 720x360 with 0 Axes>
```



In [18]:
```
df1=df.corr()
```

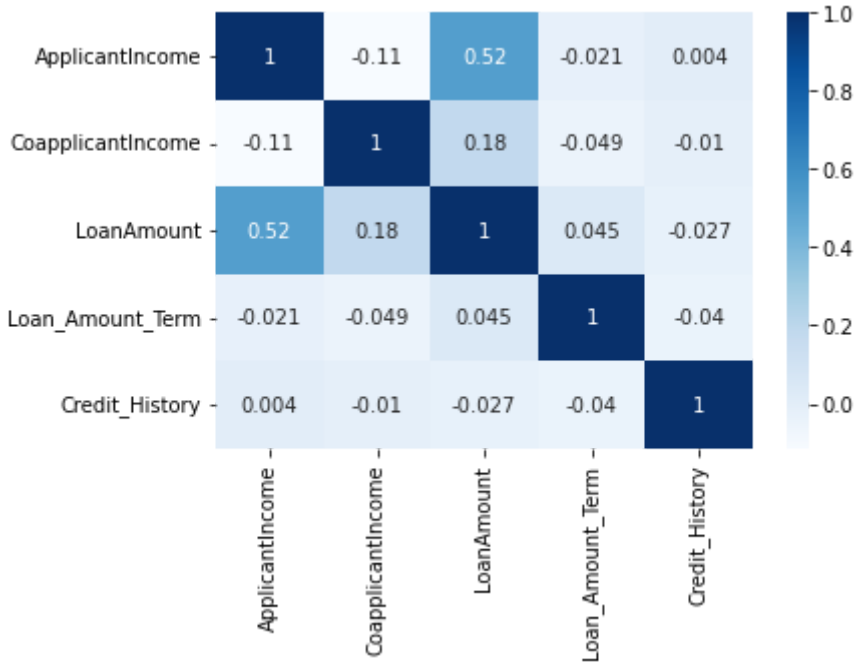
In [19]:
```
df1
```

Out[19]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cred
ApplicantIncome	1.000000	-0.114247	0.524720	-0.020953	
CoapplicantIncome	-0.114247	1.000000	0.177795	-0.048790	
LoanAmount	0.524720	0.177795	1.000000	0.045083	
Loan_Amount_Term	-0.020953	-0.048790	0.045083	1.000000	
Credit_History	0.003958	-0.010171	-0.026990	-0.040264	

In [20]:
```
sns.heatmap(df1,annot=True,cmap='Blues')
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
#REMOVING UNWANTED COLUMNS IN DATSET
```

In [ ]:
```

```

In [22]:
```
df
```

Out[22]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
976	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	
977	LP002975	Male	Yes	0	Graduate	No	4158	
978	LP002980	Male	No	0	Graduate	No	3250	
979	LP002986	Male	Yes	0	Graduate	No	5000	
980	LP002989	Male	No	0	Graduate	Yes	9200	

981 rows × 13 columns

In [23]:
```
#REMOVING CONTINUOUS FEATURES(IV'S).iv'S WITH CONTNUOUS DATA
df.drop('Dependents',axis=1,inplace=True)
df.drop('ApplicantIncome',axis=1,inplace=True)
df.drop('CoapplicantIncome',axis=1,inplace=True)
df.drop('LoanAmount',axis=1,inplace=True)
df.drop('Loan_Amount_Term',axis=1,inplace=True)
```

In [ ]:

In [24]:
```
# is used for data manipulation. It converts categorical data into dummy or indicat
df1=pd.get_dummies(df,columns=["Gender","Married","Education","Self_Employed","Prop
```

In [25]:
```
df1
```

Out[25]:

	Loan_ID	Gender_Female	Gender_Male	Married_No	Married_Yes	Education_Graduate	Edu
0	LP001002	0	1	1	0	1	
1	LP001003	0	1	0	1	1	
2	LP001005	0	1	0	1	1	
3	LP001006	0	1	0	1	0	
4	LP001008	0	1	1	0	1	
...	...	...	...	...	...	...	
976	LP002971	0	1	0	1	0	
977	LP002975	0	1	0	1	1	
978	LP002980	0	1	1	0	1	
979	LP002986	0	1	0	1	1	
980	LP002989	0	1	1	0	1	

981 rows × 16 columns

In [26]:
```
df1.drop('Gender_Female',axis=1,inplace=True)
df1.drop('Married_No',axis=1,inplace=True)
df1.drop('Education_Graduate',axis=1,inplace=True)
df1.drop('Self_Employed_No',axis=1,inplace=True)
df1.drop('Property_Area_Rural',axis=1,inplace=True)
df1.drop('Loan_Status_N',axis=1,inplace=True)
df1.drop('Credit_History_0.0',axis=1,inplace=True)
```

In [27]:
```
df1.head()
```

Out[27]:

	Loan_ID	Gender_Male	Married_Yes	Education_Not Graduate	Self_Employed_Yes	Property_Area_Semiurl
0	LP001002	1	0	0	0	
1	LP001003	1	1	0	0	
2	LP001005	1	1	0	1	
3	LP001006	1	1	1	0	
4	LP001008	1	0	0	0	

In [28]:
```
#Instantiating train_test split library from scikit learning
from sklearn.model_selection import train_test_split
```

In [29]:
```
X = df1[["Gender_Male","Married_Yes","Education_Not Graduate","Self_Employed_Yes",'
y=df1["Loan_Status_Y"].values
```

In [ ]:
```
```

In [30]:
```
#xtrain=80% of the training dataset of Independent Variables for which training has
#xtest=
#ytrain=
#ytest
```

In [61]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.4,random_state=0)
```

In [62]:
```
#Instantiating Logistic Regression from sckit learning
```

In [63]:
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
l1=LogisticRegression()
l1.fit(xtrain,ytrain)
yp1=l1.predict(xtest)
```

In [64]:
```
#Descision Tree
from sklearn.tree import DecisionTreeClassifier
d1=DecisionTreeClassifier()
d1.fit(xtrain,ytrain)
yp2=d1.predict(xtest)
```

In [65]:
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
```

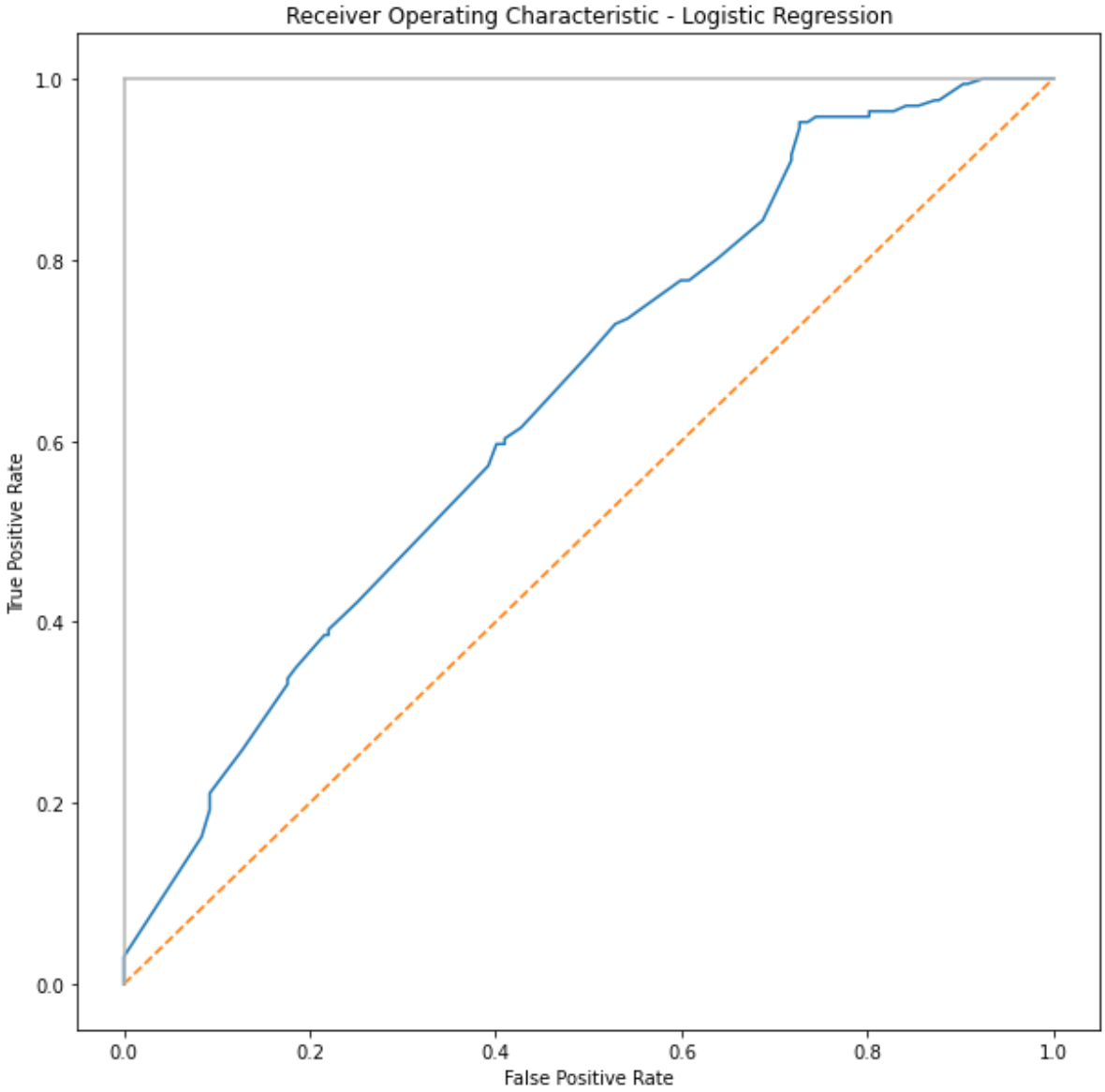
In [66]:
```
print("Logistic Regression :")
print(" accuracy is ",accuracy_score(ytest,yp1))
print(" precision score is ",precision_score(ytest,yp1))
print(" recall is ",recall_score(ytest,yp1))
print(" f1 score is ",f1_score(ytest,yp1))
print("
print(" classification report is ",classification_report(ytest,yp1))
print(" confusion matrix is ",confusion_matrix(ytest,yp1))
```

```
Logistic Regression :
 accuracy is  0.5928753180661578
 precision score is  0.5163043478260869
 recall is  0.572289156626506
 f1 score is  0.5428571428571428

 classification report is               precision    recall  f1-score   support

           0       0.66      0.61      0.63       227
           1       0.52      0.57      0.54       166

    accuracy                           0.59       393
   macro avg       0.59      0.59      0.59       393
weighted avg       0.60      0.59      0.59       393

 confusion matrix is  [[138  89]
 [ 71  95]]
```

In [67]:
```
 #Logistic Reg
y_score1 = l1.predict_proba(xtest)[:,1]
false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(ytest, y_score1)
 print('roc_auc_score for DecisionTree: ', roc_auc_score(ytest, y_score1))
```

```
roc_auc_score for DecisionTree:  0.6462501990340215
```

In [68]:
```
    plt.subplots(1, figsize=(10,10))
    plt.title('Receiver Operating Characteristic - Logistic Regression')
    plt.plot(false_positive_rate1, true_positive_rate1)
    plt.plot([0, 1], ls="--")
    plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```



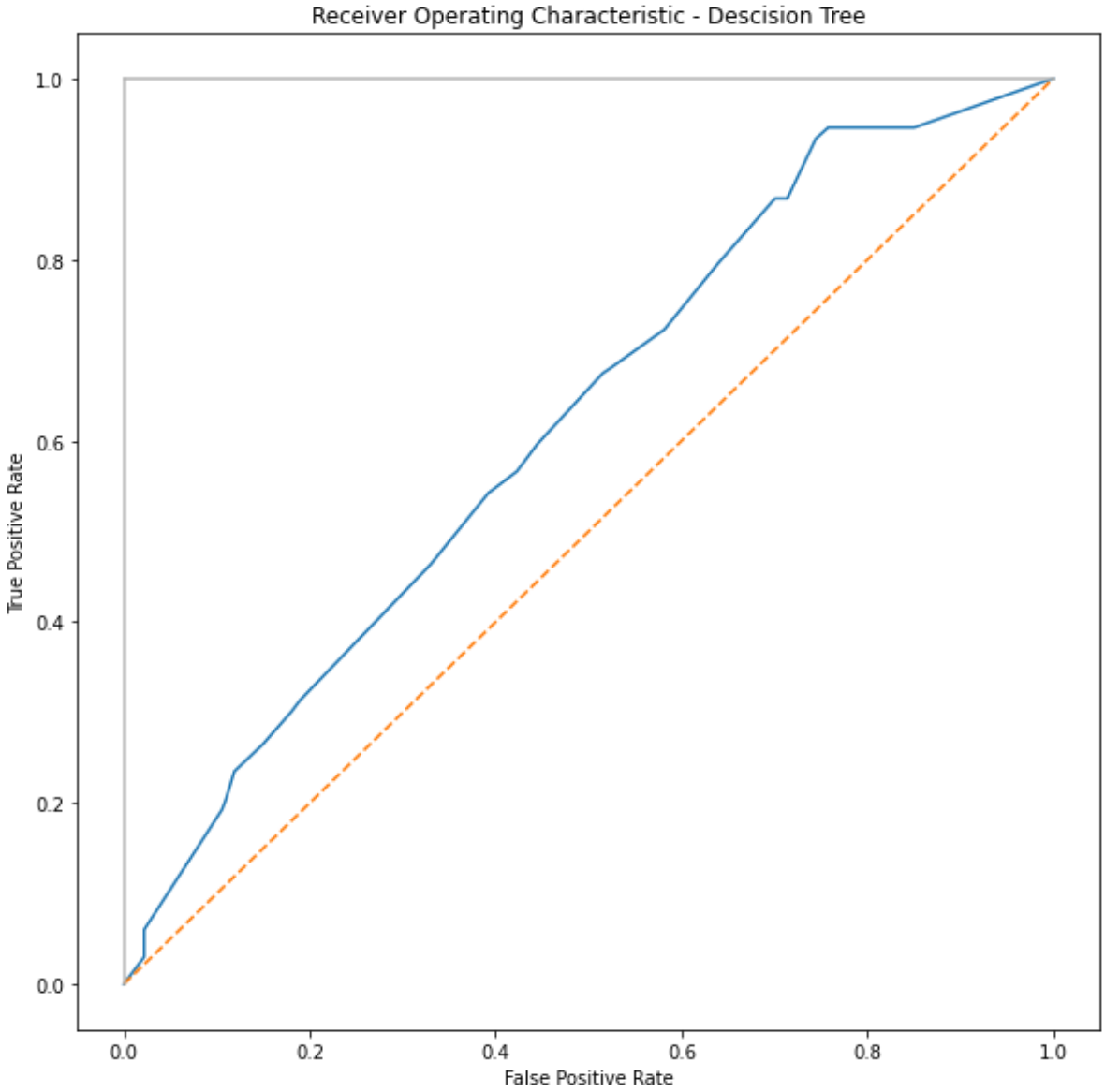
In [69]:
```
print("Decision Tree Classifier :")
print(" accuracy is ",accuracy_score(ytest,yp2))
print(" precision score is ",precision_score(ytest,yp2))
print(" recall is ",recall_score(ytest,yp2))
print(" f1 score is ",f1_score(ytest,yp2))
print("
print(" classification report is ",classification_report(ytest,yp2))
print(" confusion matrix is ",confusion_matrix(ytest,yp2))
```

```
Decision Tree Classifier :
 accuracy is  0.5725190839694656
 precision score is  0.49473684210526314
 recall is  0.5662650602409639
 f1 score is  0.5280898876404494

 classification report is               precision    recall  f1-score   support

           0       0.65      0.58      0.61       227
           1       0.49      0.57      0.53       166

    accuracy                           0.57       393
   macro avg       0.57      0.57      0.57       393
weighted avg       0.58      0.57      0.57       393

 confusion matrix is  [[131  96]
 [ 72  94]]
```

In [70]:
```
#Descision Tree
y_score2=d1.predict_proba(xtest)[:,1]
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(ytest, y_score2)
print('roc_auc_score for DecisionTree: ', roc_auc_score(ytest, y_score2))
```

```
roc_auc_score for DecisionTree:  0.6191417652990818
```

In [71]:
```
    plt.subplots(1, figsize=(10,10))
    plt.title('Receiver Operating Characteristic - Descision Tree')
    plt.plot(false_positive_rate2, true_positive_rate2)
    plt.plot([0, 1], ls="--")
    plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```



In [72]:
```
#K Nearest Neighbour
```

In [73]:
```
from sklearn.linear_model import LogisticRegression
l1=LogisticRegression()
l1.fit(xtrain,ytrain)
yp1=l1.predict(xtest)
```

In [74]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [75]:
```
#DEFAULT value is 5 number of neighbours to be used for k neigbour queries
knn5 = KNeighborsClassifier(n_neighbors = 5)
#Value changed and now only 1  neighbour to be used for k neigbour queries
knn1 = KNeighborsClassifier(n_neighbors=1)
```

In [76]:
```
knn5.fit(xtrain,ytrain)
yp3=knn5.predict(xtest)
knn1.fit(xtrain,ytrain)
yp4=knn1.predict(xtest)
```

In [77]:
```
print("K-Nearest-Neighbour  :")
print(" accuracy is ",accuracy_score(ytest,yp3))
print(" precision score is ",precision_score(ytest,yp3))
print(" recall is ",recall_score(ytest,yp3))
print(" f1 score is ",f1_score(ytest,yp3))
```

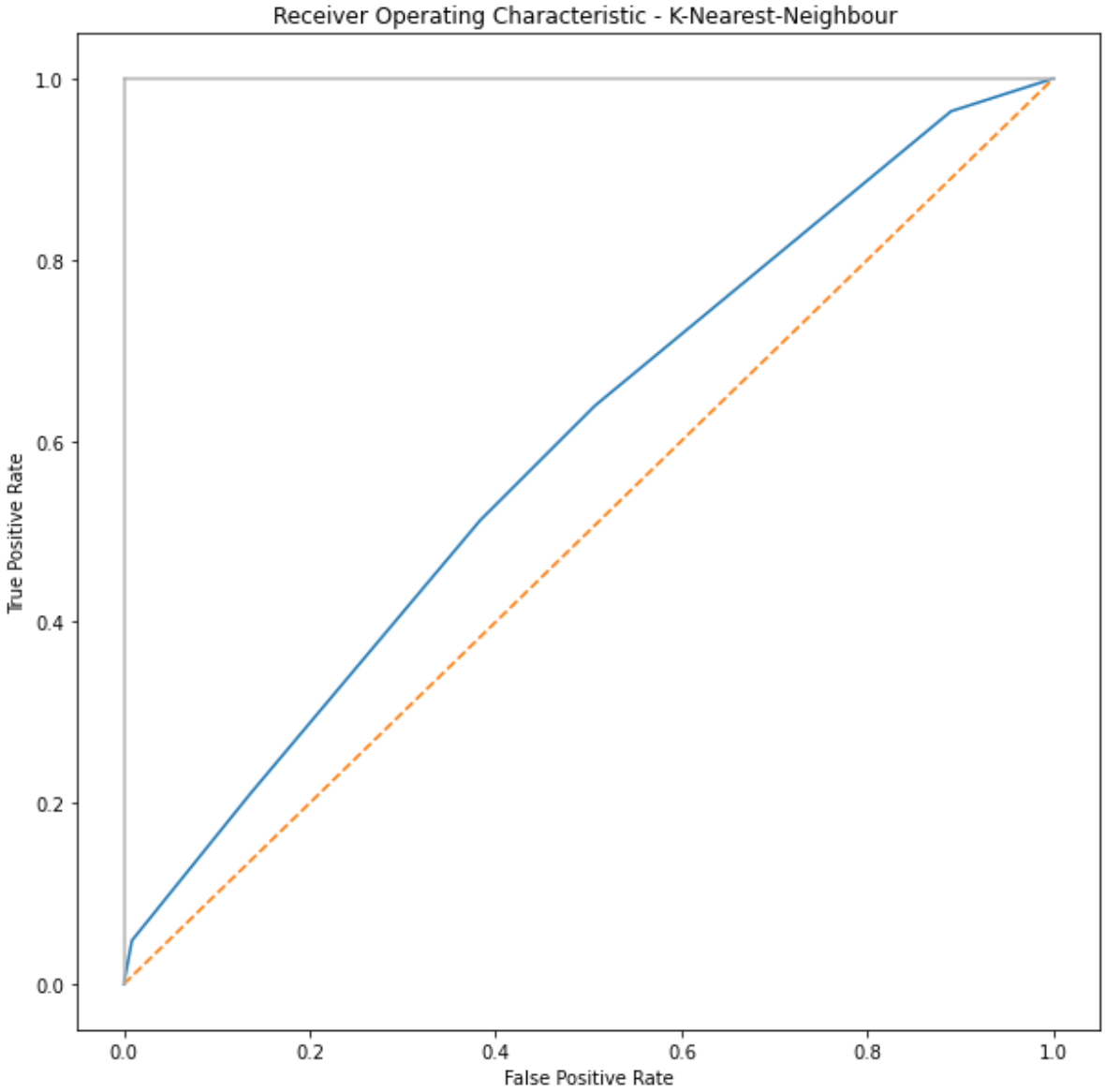
```
print("
print(" classification report is ",classification_report(ytest,yp3))
print(" confusion matrix is ",confusion_matrix(ytest,yp3))
```

```
K-Nearest-Neighbour  :
 accuracy is  0.5725190839694656
 precision score is  0.4941860465116279
 recall is  0.5120481927710844
 f1 score is  0.5029585798816567

 classification report is               precision    recall  f1-score   support

           0       0.63      0.62      0.63       227
           1       0.49      0.51      0.50       166

    accuracy                           0.57       393
   macro avg       0.56      0.56      0.56       393
weighted avg       0.57      0.57      0.57       393

 confusion matrix is  [[140  87]
 [ 81  85]]
```

In [78]:
```
y_score3=knn5.predict_proba(xtest)[:,1]
false_positive_rate3, true_positive_rate3, threshold3 = roc_curve(ytest, y_score3)
print('roc_auc_score for DecisionTree: ', roc_auc_score(ytest, y_score3))
```

```
roc_auc_score for DecisionTree:  0.5920996762379915
```

In [79]:
```
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - K-Nearest-Neighbour ')
plt.plot(false_positive_rate3, true_positive_rate3)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [164…
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [ ]:
```
```