In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df2=pd.read_csv(r'C:\Users\user\Downloads\C8_loan-test.csv')
df2
fin2=pd.read_csv(r'C:\Users\user\Downloads\C8_loan-train.csv')
fin2
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [3]:

```
df0=df2.head(10)
df=df0.fillna("30.0")
df
fin0=fin2.head(10)
fin=fin0.fillna("35.0")
fin
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
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	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	
7	LP001014	Male	Yes	3+	Graduate	No	3036	
8	LP001018	Male	Yes	2	Graduate	No	4006	
9	LP001020	Male	Yes	1	Graduate	No	12841	

In [4]:

```
df.info()
fin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            10 non-null     object 
 1   Gender             10 non-null     object 
 2   Married            10 non-null     object 
 3   Dependents         10 non-null     object 
 4   Education          10 non-null     object 
 5   Self_Employed      10 non-null     object 
 6   ApplicantIncome    10 non-null     int64  
 7   CoapplicantIncome  10 non-null     int64  
 8   LoanAmount         10 non-null     float64
 9   Loan_Amount_Term   10 non-null     float64
 10  Credit_History     10 non-null     object 
 11  Property_Area      10 non-null     object 
dtypes: float64(2), int64(2), object(8)
memory usage: 1.1+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            10 non-null     object 
 1   Gender             10 non-null     object 
 2   Married            10 non-null     object 
 3   Dependents         10 non-null     object 
 4   Education          10 non-null     object 
 5   Self_Employed      10 non-null     object 
 6   ApplicantIncome    10 non-null     int64  
 7   CoapplicantIncome  10 non-null     float64
 8   LoanAmount         10 non-null     object 
 9   Loan_Amount_Term   10 non-null     float64
 10  Credit_History     10 non-null     float64
 11  Property_Area      10 non-null     object 
 12  Loan_Status        10 non-null     object 
dtypes: float64(3), int64(1), object(9)
memory usage: 1.1+ KB
```

In [5]:

```
df.describe()
fin.describe()
```

Out[5]:

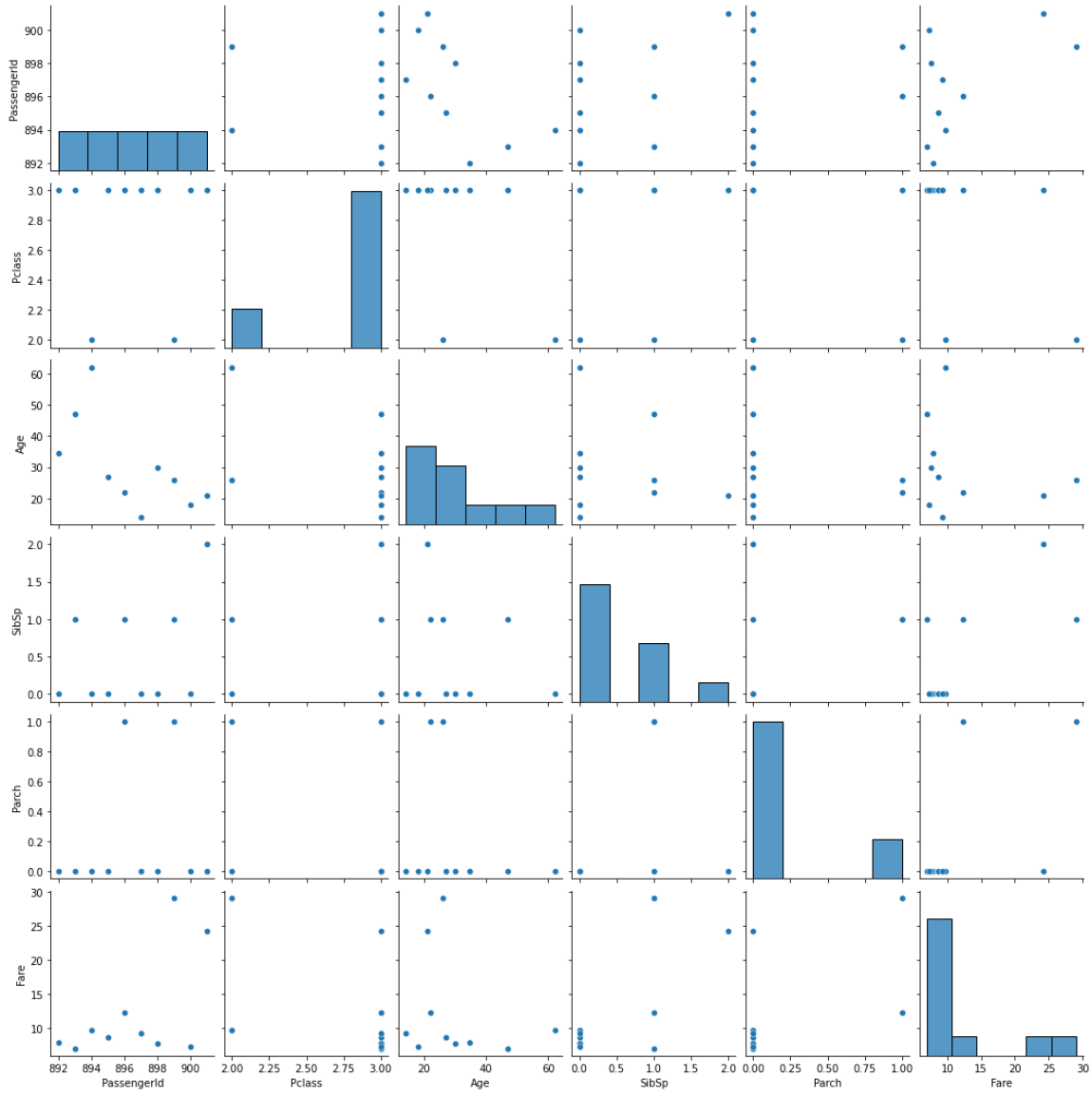
	ApplicantIncome	CoapplicantIncome	Loan_Amount_Term	Credit_History
count	10.000000	10.000000	10.0	10.000000
mean	4964.800000	2457.600000	360.0	0.900000
std	3079.278047	3270.009147	0.0	0.316228
min	2333.000000	0.000000	360.0	0.000000
25%	3009.000000	377.000000	360.0	1.000000
50%	4294.500000	1521.000000	360.0	1.000000
75%	5741.000000	2467.500000	360.0	1.000000
max	12841.000000	10968.000000	360.0	1.000000

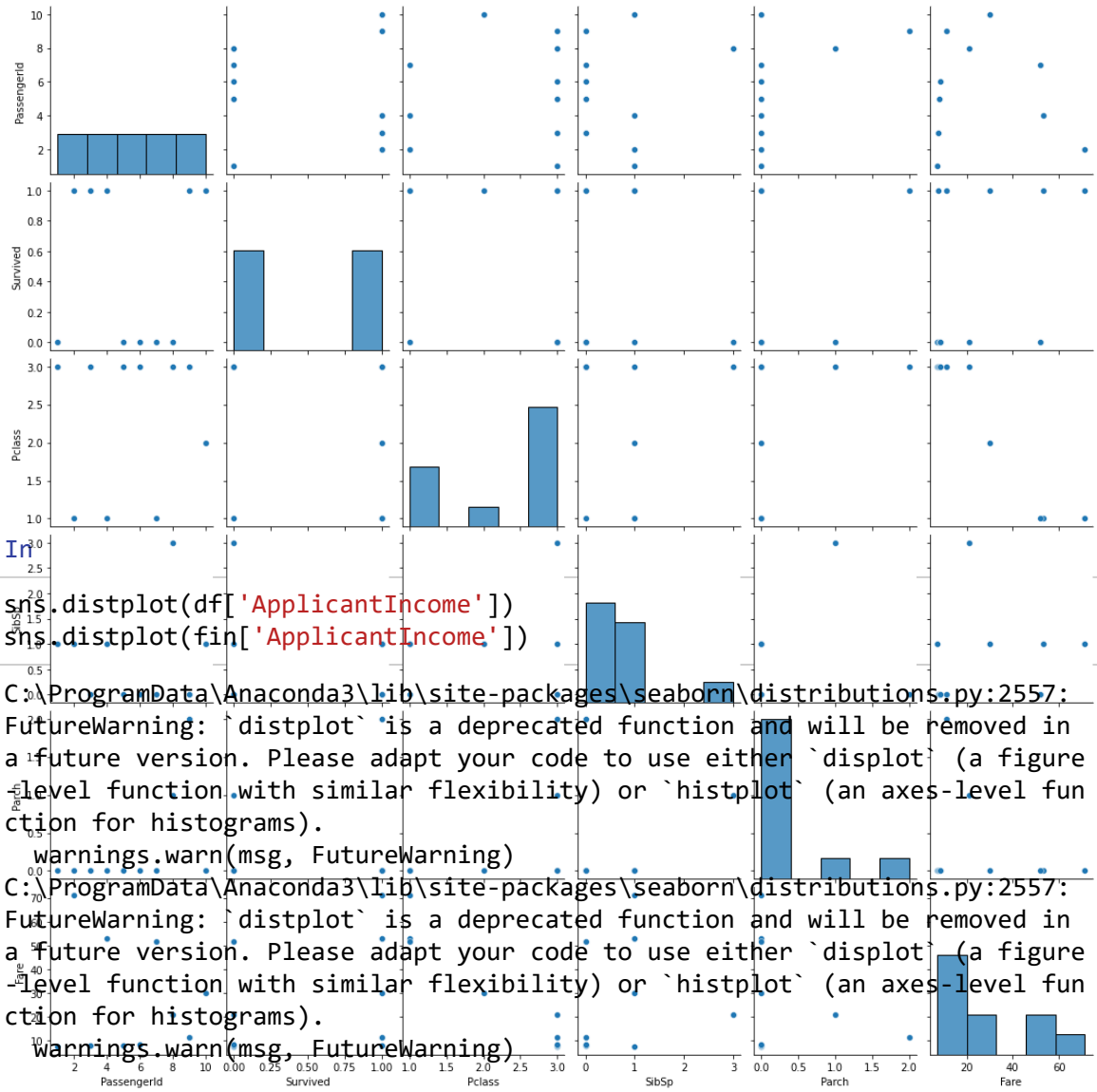
In [7]:

```
sns.pairplot(df)
sns.pairplot(fin)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2918b32d340>
```





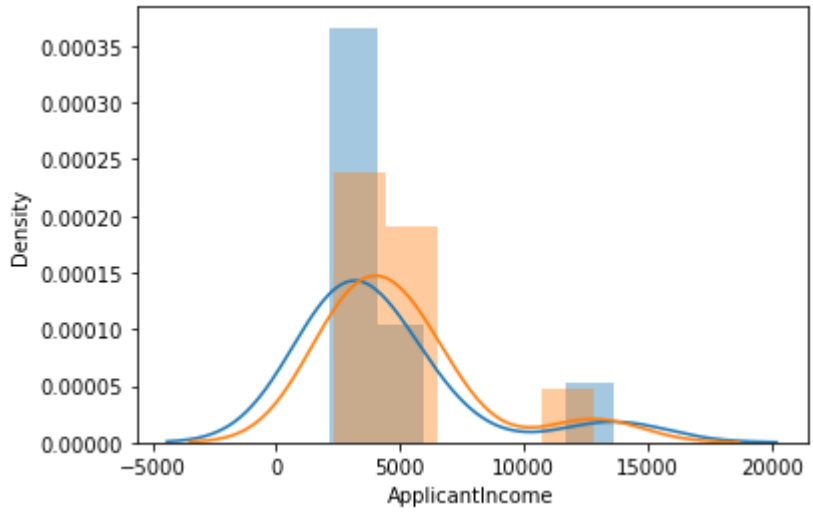
In

```
sns.distplot(df['ApplicantIncome'])
sns.distplot(fin['ApplicantIncome'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[7]:

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



In [8]:

```
df.columns
fin.columns
```

Out[8]:

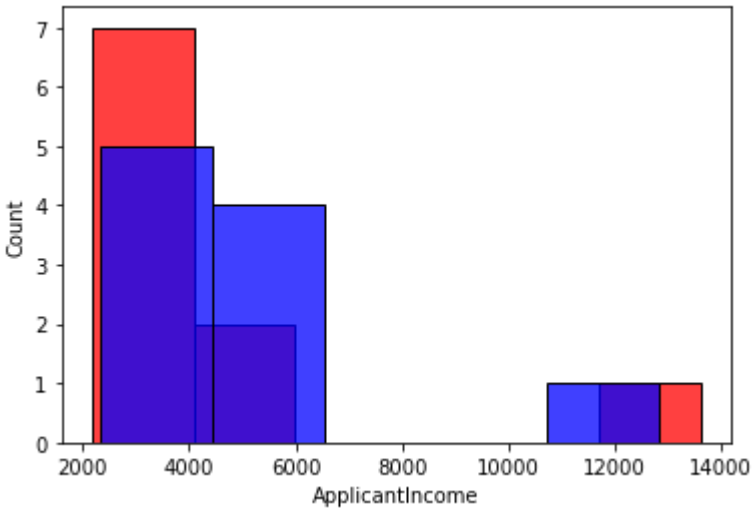
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [9]:

```
sns.histplot(df["ApplicantIncome"],color='red')
sns.histplot(fin["ApplicantIncome"],color='blue')
```

Out[9]:

```
<AxesSubplot:xlabel='ApplicantIncome', ylabel='Count'>
```



In [12]:

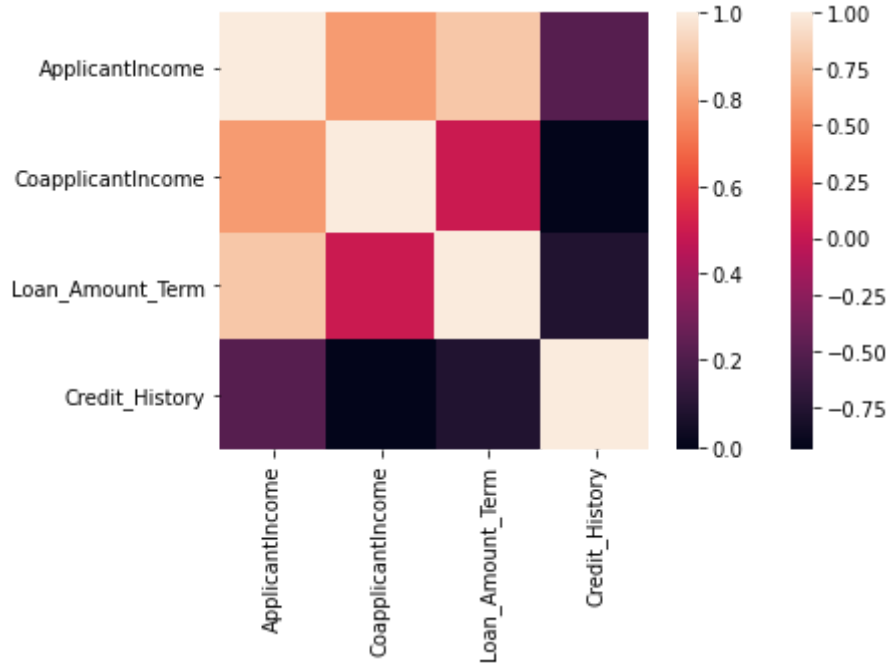
```
df1=df[['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area']]
fin1=fin[['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status']]
```

In [13]:

```
sns.heatmap(df1.corr())
sns.heatmap(fin1.corr())
```

Out[13]:

<AxesSubplot:>



In [14]:

```
x=df1[[ 'ApplicantIncome', 'CoapplicantIncome', 'Loan_Amount_Term', 'Credit_History']]
y=fin1[[ 'ApplicantIncome', 'CoapplicantIncome', 'Loan_Amount_Term', 'Credit_History']]
```

In [15]:

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

In [16]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[17]:

```
LinearRegression()
```

In [18]:

```
print(lr.intercept_)
```

```
[ 7644.40139783 10895.42140038   360.           11.07030758]
```

In [19]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[19]:

	0	1	2	3
0	-0.093196	-0.018138	-9.687834	-42.778210
1	-0.263048	-0.234698	-24.287564	45.727672
2	0.000000	0.000000	0.000000	0.000000
3	-0.000270	0.000366	-0.026533	-0.026727

In [20]:

```
print(lr.score(x_test,y_test))
```

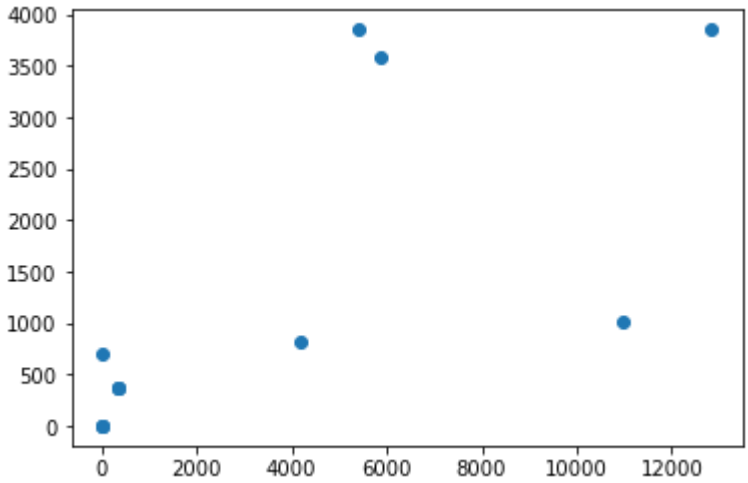
```
-0.3406346403462329
```

In [21]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x23f0086e0d0>
```



In [22]:

```
lr.score(x_test,y_test)
```

Out[22]:

```
-0.3406346403462329
```

In [23]:

```
lr.score(x_train,y_train)
```

Out[23]:

```
0.4760508070555572
```

In [24]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]:

```
Ridge(alpha=10)
```

In [26]:

```
rr.score(x_test,y_test)
```

Out[26]:

```
-0.3436647603650187
```

In [27]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.0, tolerance:
0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[27]:

```
Lasso(alpha=10)
```

In [28]:

```
la.score(x_test,y_test)
```

Out[28]:

```
-0.3419641974907315
```

Elastic Net

In [29]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.0, tolerance:
0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[29]:

```
ElasticNet()
```

In [30]:

```
print(en.coef_)
```

```
[[-8.76032173e-02 -2.49483923e-02 -9.18220185e+00 -4.18999775e+01]
 [-2.65042842e-01 -2.30256804e-01 -2.44647573e+01  4.51255162e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [-5.92017184e-05  1.52772784e-04 -7.44148759e-03 -0.00000000e+00]]
```

In [31]:

```
print(en.intercept_)
```

```
[7.44562610e+03 1.09650026e+04 3.60000000e+02 3.56367034e+00]
```

In [32]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[[3.82309909e+03 8.41058926e+02 3.60000000e+02 1.27935155e+00]
 [3.82800960e+03 1.01409631e+03 3.60000000e+02 1.10930536e+00]
 [3.59704306e+03 6.86770406e+02 3.60000000e+02 5.46100977e-01]]
```

In [33]:

```
print(en.score(x_test,y_test))
```

```
-0.34178623915447015
```

Logistic Regression

In [34]:

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

In [47]:

```
feature_matrix=df1[[ 'ApplicantIncome','CoapplicantIncome']]
target_vector = df1[['ApplicantIncome']]
```

In [48]:

```
feature_matrix.shape
```

Out[48]:

```
(10, 2)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(10, 1)
```

In [50]:

```
from sklearn.preprocessing import StandardScaler
```

In [51]:

```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [52]:

```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  return f(*args, **kwargs)
```

Out[52]:

```
LogisticRegression()
```

In [55]:

```
observation=df1[['ApplicantIncome','CoapplicantIncome']]
```

In [56]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
[13633 13633 13633 13633 13633  2165 13633 13633 13633 13633]
```

In [57]:

```
logr.classes_
```

Out[57]:

```
array([ 2165,  2226,  2340,  2400,  3076,  3276,  3881,  5000,  5720,
       13633], dtype=int64)
```

In [58]:

```
logr.predict_proba(observation)[0][0]
```

Out[58]:

```
0.0
```

In [59]:

```
logr.predict_proba(observation)[0][1]
```

Out[59]:

```
0.0
```

Evaluation Metrics

In [60]:

```
from sklearn import metrics
```

In [61]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-61-4b3ebc69c71d> in <module>
----> 1 print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,pr
ediction))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_absolute_error(y_true, y_pred, sample_weight, multioutput)
    180     0.85...
    181     """
--> 182     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    183         y_true, y_pred, multioutput)
    184     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

In [62]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-62-a8cbb0b7f78a> in <module>
----> 1 print("Mean Squared Error:",metrics.mean_squared_error(y_test,pred
iction))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_squared_error(y_true, y_pred, sample_weight, multioutput, squared)
    333     0.825...
    334     """
--> 335     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    336         y_true, y_pred, multioutput)
    337     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

In [63]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-63-c32fd676945a> in <module>
----> 1 print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_erro
r(y_test,prediction)))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_squared_error(y_true, y_pred, sample_weight, multioutput, squared)
    333     0.825...
    334     """
--> 335     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    336         y_true, y_pred, multioutput)
    337     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

Model Saving

In [64]:

```
import pickle
```

In [65]:

```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```