OUTSTANDING PROJECT 1

Mudit Mathur – mm7692@srmist.edu.in

PROBLEM: LOAN ELIGIBILITY AND AMOUNT PREDICTION

DESCRIPTION OF DATASET:

Dataset includes loan_id, gender, married, dependents, Education, Applicant Income, Credit History, Loan_status, etc

Total of 614 Entries and 13 Features(Columns).

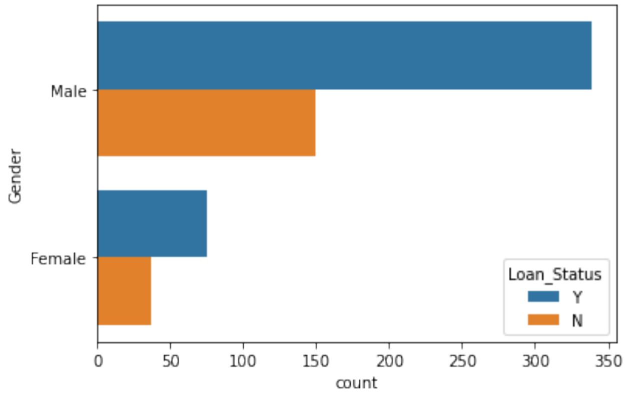
LIBRARIES USED:

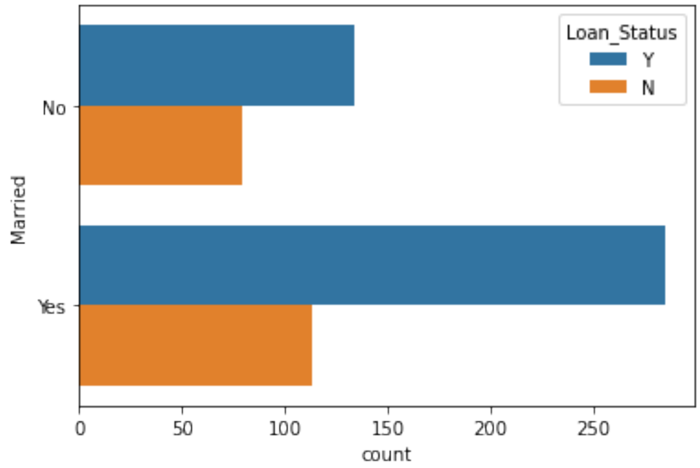
Pandas, Numpy, Seaborn, Matplotlib, Sci-kit learn

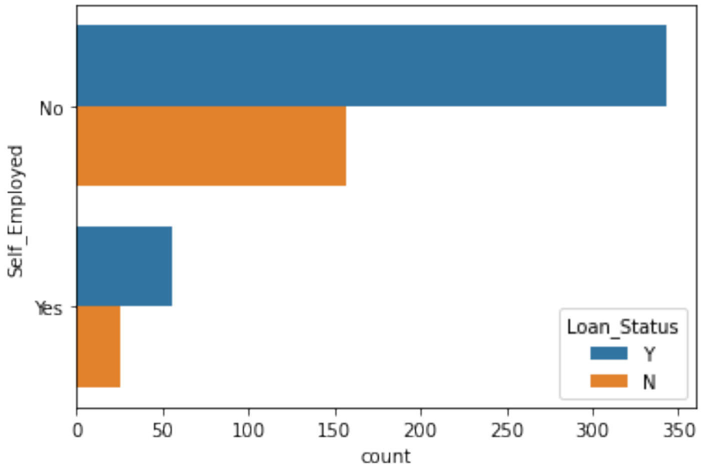
DATA CLEANING:

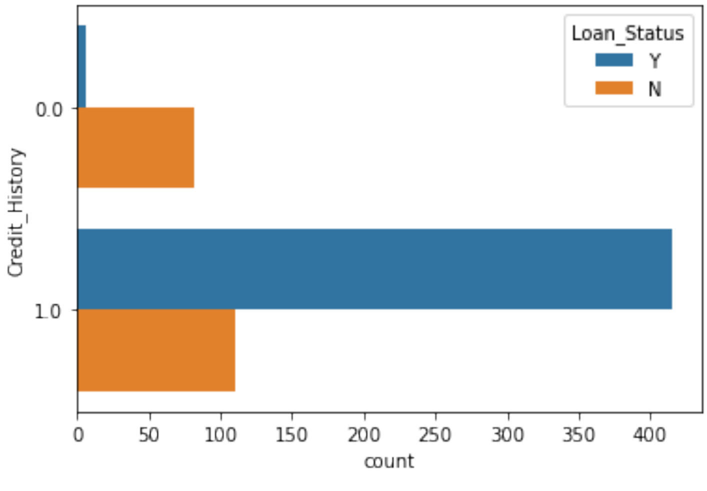
1. Filling NULL Values: Inserting Median values in the NULL values.
2. Converting Gender into Numerical Data : Encoding the gender values using get_dummies
3. Converting Categorical Features into Numerical: Converting Married, Self_employed, Dependents into Numerical Data
4. Removing Redundant Columns using Correlation Matrix(Heatmap)

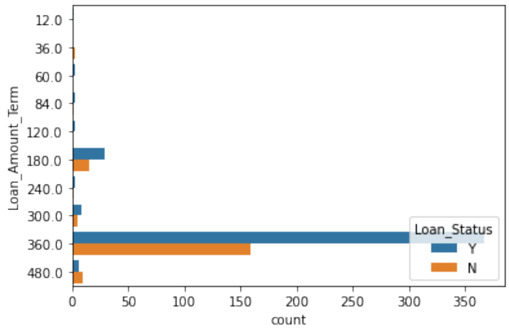
EXPLORATORY DATA ANALYSIS (EDA):



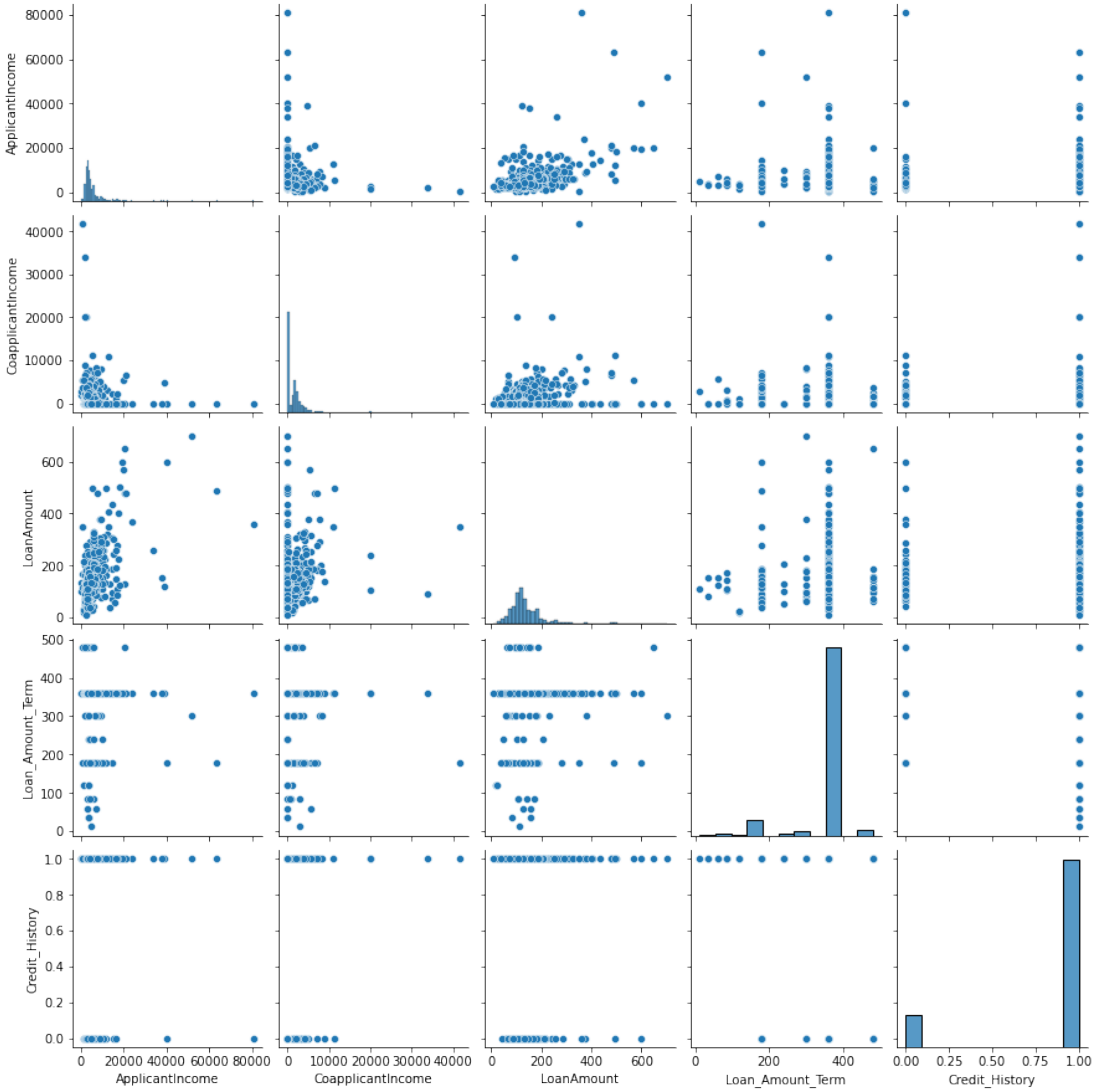

Loan_Status
Y
N
Married
No
Yes
0
50
100
150
200
250
count


Self_Employed
No
Yes
Loan_Status
Y
N
0
50
100
150
200
250
300
350
count


Loan_Status
Y
N
Credit_History
0.0
1.0
0
50
100
150
200
250
300
350
400
count


Loan_Amount_Term
12.0
36.0
60.0
84.0
120.0
180.0
240.0
300.0
360.0
480.0
Loan_Status
Y
N
0
50
100
150
200
250
300
350
count

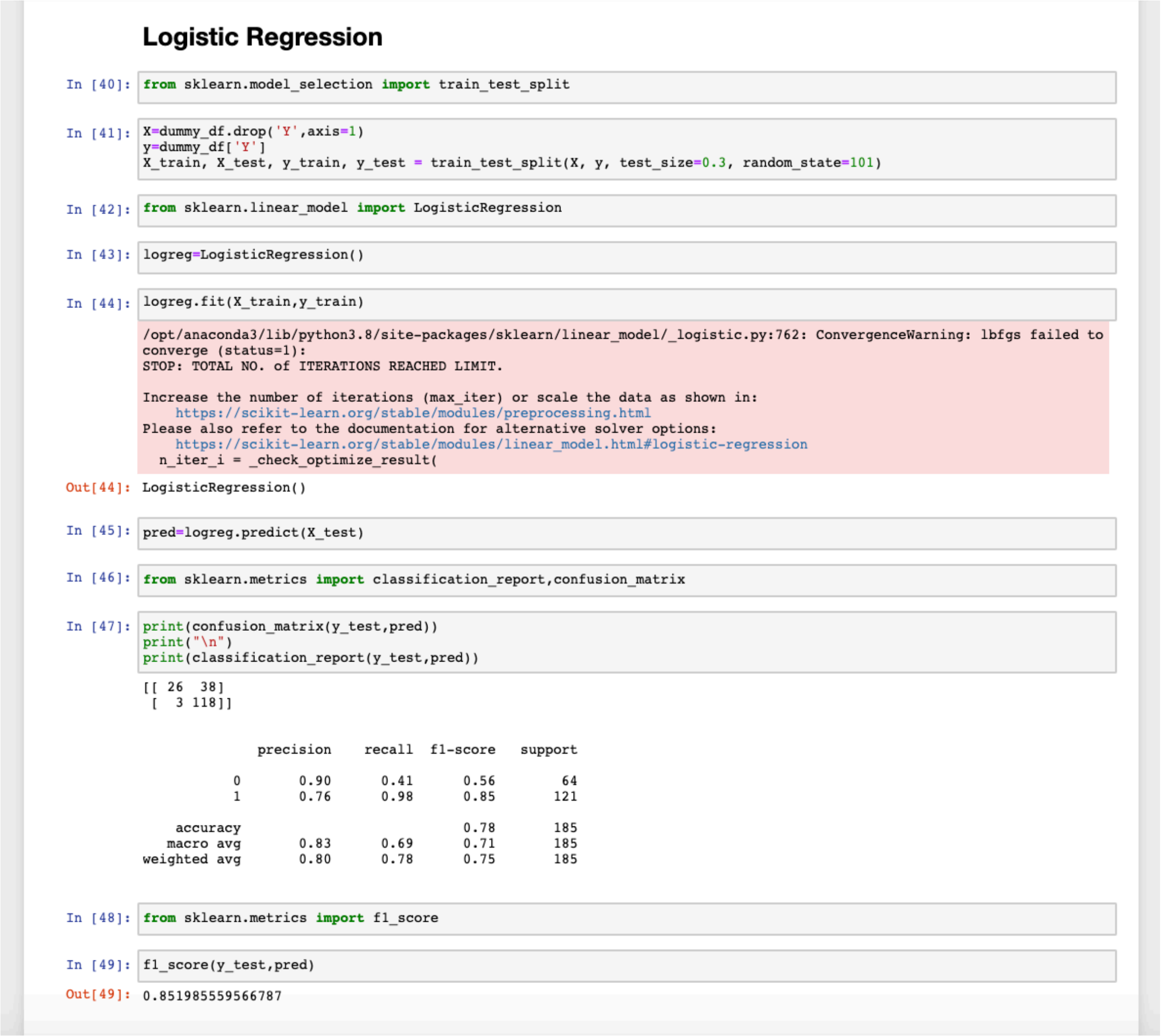
PAIRPLOT



MACHINE LEARNING AND ALGORITHMS:

1. Logistic Regression
2. Random Forest
3. Support Vector Machine
4. Linear Regression (Loan Amount Pred.) – GRIDSEARCH CV
5. SVM Regression (Loan Amount Pred.)

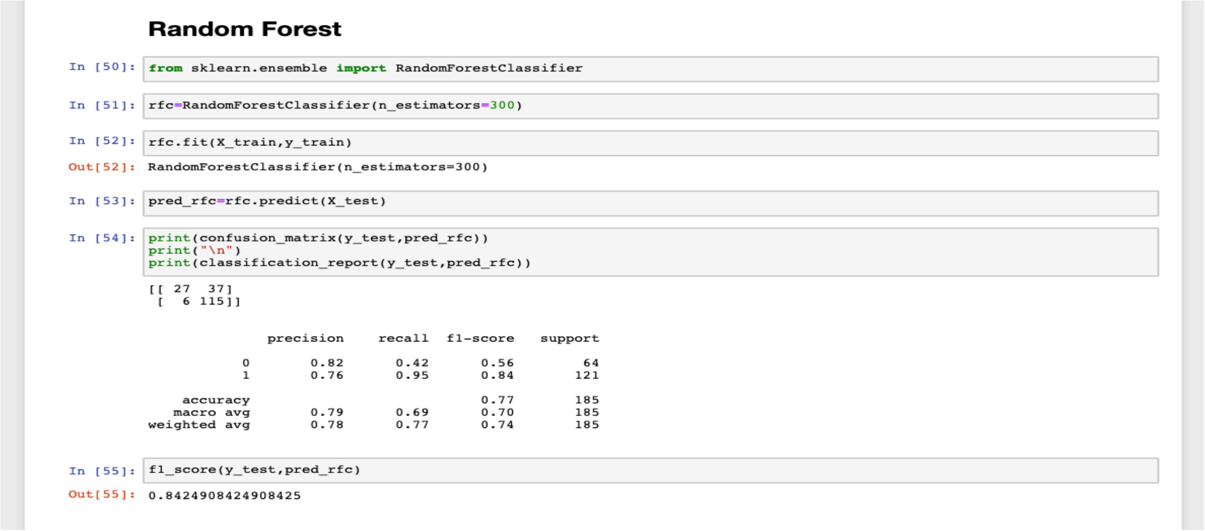
Logistic Regression (f1_Score=85.19%)



Logistic Regression

```
In [40]: from sklearn.model_selection import train_test_split

In [41]: X=dummy_df.drop('Y',axis=1)
         y=dummy_df['Y']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

In [42]: from sklearn.linear_model import LogisticRegression

In [43]: logreg=LogisticRegression()

In [44]: logreg.fit(X_train,y_train)

         /opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to
         converge (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(

Out[44]: LogisticRegression()

In [45]: pred=logreg.predict(X_test)

In [46]: from sklearn.metrics import classification_report,confusion_matrix

In [47]: print(confusion_matrix(y_test,pred))
         print("\n")
         print(classification_report(y_test,pred))

         [[ 26  38]
          [  3 118]]


                       precision    recall  f1-score   support

                    0       0.90      0.41      0.56        64
                    1       0.76      0.98      0.85       121

             accuracy                           0.78       185
            macro avg       0.83      0.69      0.71       185
         weighted avg       0.80      0.78      0.75       185


In [48]: from sklearn.metrics import f1_score

In [49]: f1_score(y_test,pred)

Out[49]: 0.851985559566787
```

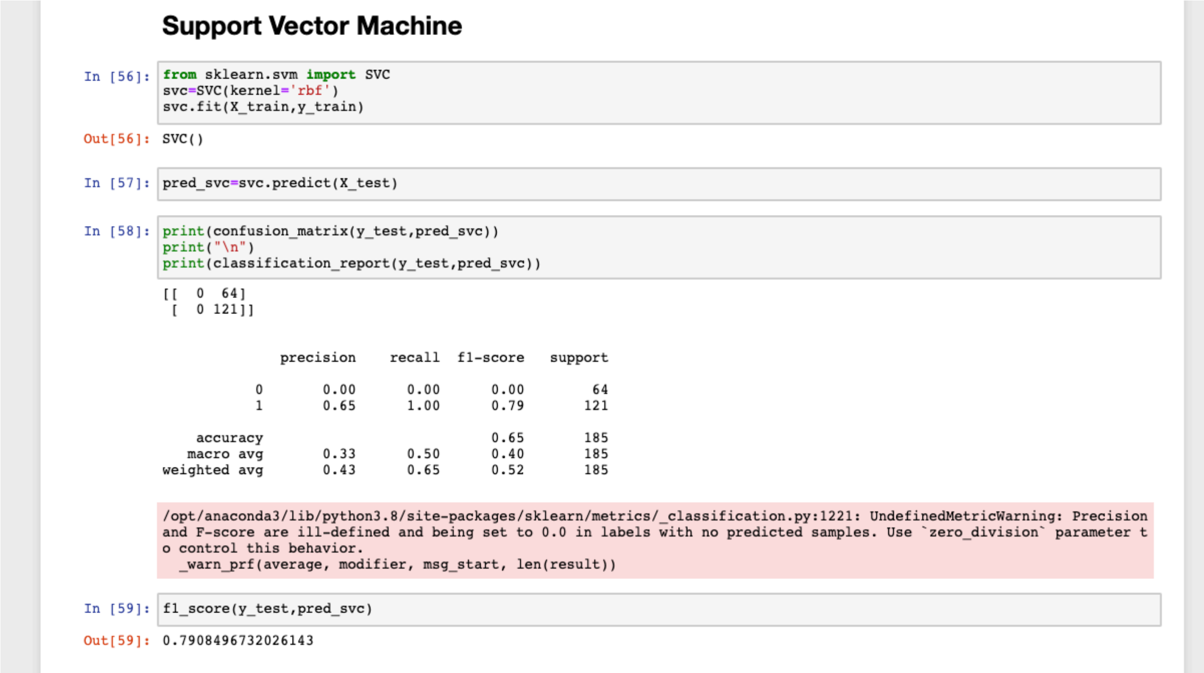
Random Forest (f1_Score=84.24%)



Random Forest

```
In [50]: from sklearn.ensemble import RandomForestClassifier

In [51]: rfc=RandomForestClassifier(n_estimators=300)

In [52]: rfc.fit(X_train,y_train)

Out[52]: RandomForestClassifier(n_estimators=300)

In [53]: pred_rfc=rfc.predict(X_test)

In [54]: print(confusion_matrix(y_test,pred_rfc))
         print("\n")
         print(classification_report(y_test,pred_rfc))

         [[ 27  37]
          [  6 115]]


                       precision    recall  f1-score   support

                    0       0.82      0.42      0.56        64
                    1       0.76      0.95      0.84       121

             accuracy                           0.77       185
            macro avg       0.79      0.69      0.70       185
         weighted avg       0.78      0.77      0.74       185


In [55]: f1_score(y_test,pred_rfc)

Out[55]: 0.8424908424908425
```

SUPPORT VECTOR MACHINE(f1_Score=79.08%)

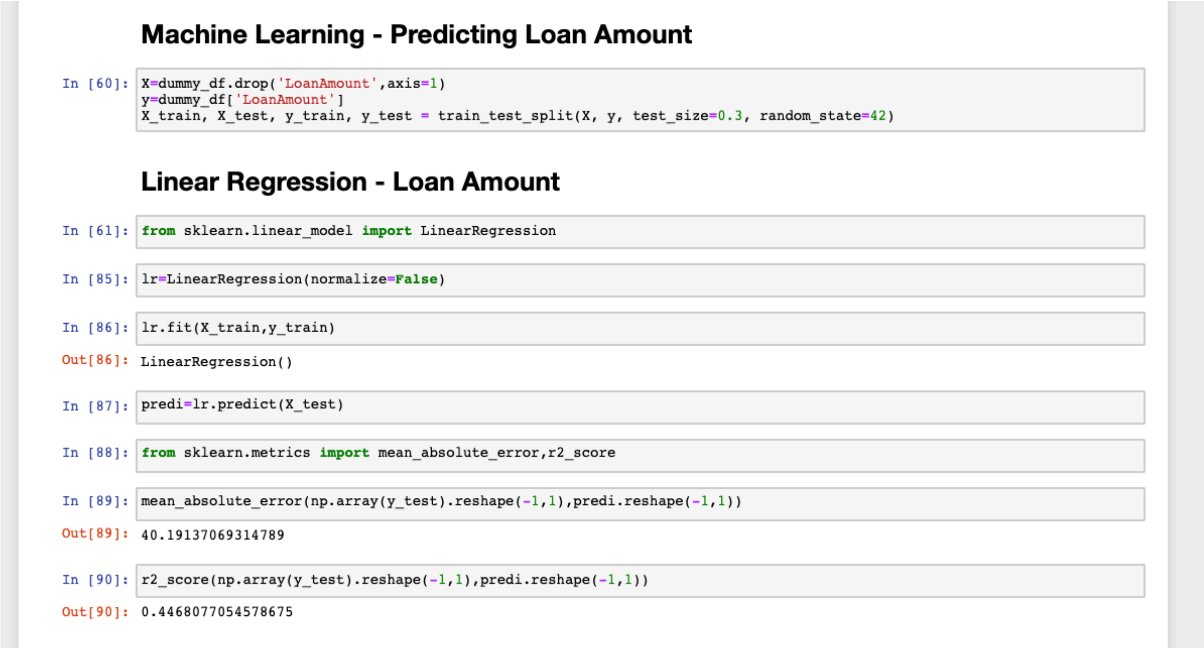


Support Vector Machine

```
In [56]: from sklearn.svm import SVC
         svc=SVC(kernel='rbf')
         svc.fit(X_train,y_train)

Out[56]: SVC()

In [57]: pred_svc=svc.predict(X_test)

In [58]: print(confusion_matrix(y_test,pred_svc))
         print("\n")
         print(classification_report(y_test,pred_svc))

         [[  0  64]
          [  0 121]]


                       precision    recall  f1-score   support

                    0       0.00      0.00      0.00        64
                    1       0.65      1.00      0.79       121

             accuracy                           0.65       185
            macro avg       0.33      0.50      0.40       185
         weighted avg       0.43      0.65      0.52       185

         /opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1221: UndefinedMetricWarning: Precision
         and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
         o control this behavior.
           _warn_prf(average, modifier, msg_start, len(result))

In [59]: f1_score(y_test,pred_svc)

Out[59]: 0.7908496732026143
```

PREDICTING LOAN AMOUNT

LINEAR REGRESSION(r2_Score=44.68%)



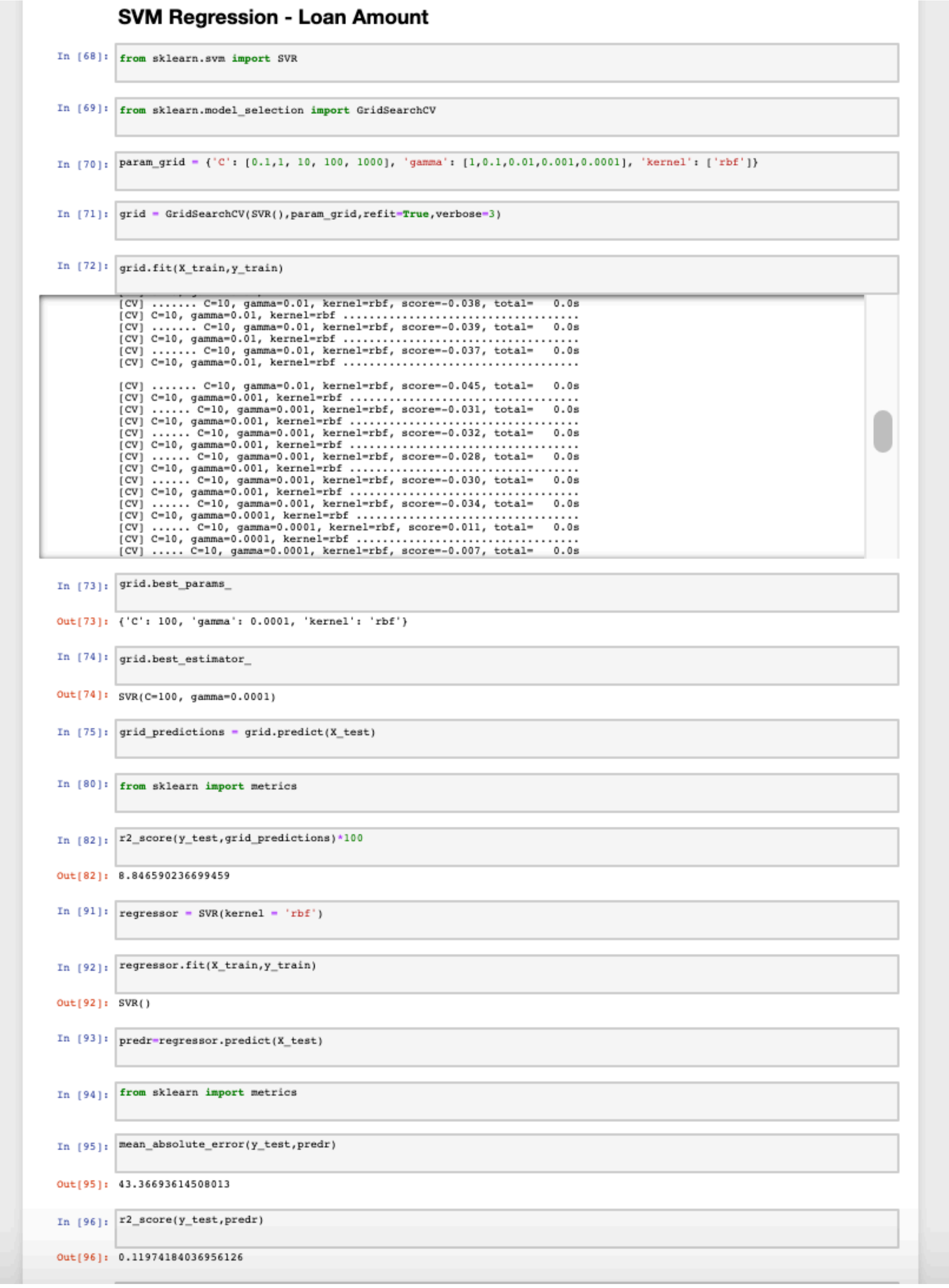
Machine Learning - Predicting Loan Amount

```
In [60]: X=dummy_df.drop('LoanAmount',axis=1)
         y=dummy_df['LoanAmount']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Linear Regression - Loan Amount

```
In [61]: from sklearn.linear_model import LinearRegression

In [85]: lr=LinearRegression(normalize=False)

In [86]: lr.fit(X_train,y_train)

Out[86]: LinearRegression()

In [87]: predi=lr.predict(X_test)

In [88]: from sklearn.metrics import mean_absolute_error,r2_score

In [89]: mean_absolute_error(np.array(y_test).reshape(-1,1),predi.reshape(-1,1))

Out[89]: 40.19137069314789

In [90]: r2_score(np.array(y_test).reshape(-1,1),predi.reshape(-1,1))

Out[90]: 0.4468077054578675
```

SVR (r2_Score=11.97%)



SVM Regression - Loan Amount

```
In [68]: from sklearn.svm import SVR

In [69]: from sklearn.model_selection import GridSearchCV

In [70]: param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf']}

In [71]: grid = GridSearchCV(SVR(),param_grid,refit=True,verbose=3)

In [72]: grid.fit(X_train,y_train)

         [CV] ....... C=10, gamma=0.01, kernel=rbf, score=-0.038, total=   0.0s
         [CV] C=10, gamma=0.01, kernel=rbf ..................................
         [CV] ....... C=10, gamma=0.01, kernel=rbf, score=-0.039, total=   0.0s
         [CV] C=10, gamma=0.01, kernel=rbf ..................................
         [CV] ....... C=10, gamma=0.01, kernel=rbf, score=-0.037, total=   0.0s
         [CV] C=10, gamma=0.01, kernel=rbf ..................................

         [CV] ....... C=10, gamma=0.01, kernel=rbf, score=-0.045, total=   0.0s
         [CV] C=10, gamma=0.001, kernel=rbf .................................
         [CV] ...... C=10, gamma=0.001, kernel=rbf, score=-0.031, total=   0.0s
         [CV] C=10, gamma=0.001, kernel=rbf .................................
         [CV] ...... C=10, gamma=0.001, kernel=rbf, score=-0.032, total=   0.0s
         [CV] C=10, gamma=0.001, kernel=rbf .................................
         [CV] ...... C=10, gamma=0.001, kernel=rbf, score=-0.028, total=   0.0s
         [CV] C=10, gamma=0.001, kernel=rbf .................................
         [CV] ...... C=10, gamma=0.001, kernel=rbf, score=-0.030, total=   0.0s
         [CV] C=10, gamma=0.001, kernel=rbf .................................
         [CV] ...... C=10, gamma=0.001, kernel=rbf, score=-0.034, total=   0.0s
         [CV] C=10, gamma=0.0001, kernel=rbf ................................
         [CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.011, total=   0.0s
         [CV] C=10, gamma=0.0001, kernel=rbf ................................
         [CV] ..... C=10, gamma=0.0001, kernel=rbf, score=-0.007, total=   0.0s

In [73]: grid.best_params_

Out[73]: {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}

In [74]: grid.best_estimator_

Out[74]: SVR(C=100, gamma=0.0001)

In [75]: grid_predictions = grid.predict(X_test)

In [80]: from sklearn import metrics

In [82]: r2_score(y_test,grid_predictions)*100

Out[82]: 8.846590236699459

In [91]: regressor = SVR(kernel = 'rbf')

In [92]: regressor.fit(X_train,y_train)

Out[92]: SVR()

In [93]: predr=regressor.predict(X_test)

In [94]: from sklearn import metrics

In [95]: mean_absolute_error(y_test,predr)

Out[95]: 43.36693614508013

In [96]: r2_score(y_test,predr)

Out[96]: 0.11974184036956126
```

CONCLUSION:

Best Machine Learning Model for the dataset is **Logistic Regression** (For Predicting Eligibility) and **Linear Regression** (For Loan Amount Prediction).