Step :1

```
import pandas as pd
import matplotlib.pyplot as plt.
train_data= pd.read_csv('train_loan.csv')
train_data.head()
train_data.shape
train_data.columns
train_data.dtypes
train_data.info()
train_data.value_counts
train_data.Dependants.value_counts()
#Replace numbers as string by integer in 'Dependants' columns
def string(x):
    if x == '0':
        return 'bad'
    elif x=='1':
        return 'average'
    elif x == '2':
        return 'good'
    else:
        return 'excellent'
train_data['Dependents'] = train_data['Dependents'].apply(string)
train_data.isna().sum()
```

Lab7. Loan Approval Classification using SVM

Objectives
In this lab, you will build a classification model to classify the loan applicants into eligible applicants or not eligible applicants using Support Vector Machine.

Learning Outcomes
After completing this lab, you will be able to

- Apply data cleaning methods
- Perform EDA and understand who got their loans approved
- Do feature engineering with One Hot Encoding
- Create LinearSVC model, train and predict on the data
- Print accuracy, confusion matrix and classification report
- Compare LinearSVC model with SVC and SGDClassifier models

Business Use Case
Heber Housing Finance deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. However doing this manually takes a lot of time. Hence, it wants you to automate the loan approval process (real time) based on customer information. So you should identify all features and build a model that enable the company to approve the load application or not.

The dataset contains the details of 614 loan applicants, where each applicant is described with 12 features. Loan_Status is the target feature (ie., dependent variable) and all others are independent variables.

Step1. [Understand Data]. Using Pandas, import **"train_loan.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Data Cleaning]

- Replace numbers as string by integer in "Dependents" column
- Fill missing data in categorical columns (Gender, Married, Dependents, Education, Self_Employed, Credit_History) by its mode value
- Handle missing values in numberical columns
- Drop Loan_ID column

Step3. [OPTIONAL: Exploratory Data Analysis - Who got their loan approved]

Draw count plot for

- Married?
- Dependants?
- Graduates?
- Self-employed?

Step4. [Extract X and y] from the dataframe

Step5. [One Hot Encoding]

Perform OHE on categorical columns, use this method: X = pd.get_dummies(X)

Step6. [Model Building]

- Split X and y for training and testing
- Using StandardScaler, fit_transform on X_train and transform on X_test values
- create LinearSVC model, train and test
- print accuracy value
- Print confusion matrix between y_test and y_pred
- Print classification_report

Categorical

```
train_data['Gender'].fillna(train_data['Gender'].mode()[0],
                            inplace=True)
train_data['Married'].fillna(train_data['Married'].mode()[0],
                             inplace=True)
train_data['Dependents'].fillna(train_data['Dependents'].
                                mode()[0], inplace=True)
train_data['Loan_Amount_Term'].fillna(train_data['Loan_Amount_Term'].
                                      mode()[0], inplace=True
train_data['Credit_History'].fillna(train_data['Credit_History'].
                                    mode()[0], inplace=True)
train_data['Self_Employed'].fillna(train_data['Self_Employed'].
                                   mode()[0], inplace=True)
```

numerical

```
train_data['Loan Amount'].fillna(train_data['Loan Amount'].
train_data = train_data.drop(['Loan_ID'], axis=1) mean(), inplace=True)
plt.subplot(231)
train_data['Married'].value_counts().plot(kind='bar',
                                          title='Gender',
plt.subplot(232)                          figsize=(20,10))
train_data['Dependents'].value_counts().plot(kind='bar',
                                             title='Dependents')
plt.subplot(233)
train_data['Education'].value_counts().plot(kind='bar',
                                            title='Education')
plt.subplot(234)
train_data['Self_Employed'].value_counts().
                                    plot(kind='bar',
plt.show()
```

Step7. [Performance Comparisons]

- Compare the performance of LinearSVC against LogisticRegression
- Compare the performance of LinearSVC against SGDClassifier
- Compare LinearSVC against SVC with various kernels such as 'linear', 'poly', 'rbf' and 'sigmoid'
- Interpret the results.

```
X = train_data.drop(['Loan_status'],axis=1)
y = train_data.Loan_status

import warnings
warnings.filterwarnings('ignore')
X = pd.get_dummies(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                   test_size=0.3, random_state=0)

from sklearn.preprocessing import StandardScaler
Scale = StandardScaler()
ss = Scale.fit_transform(X_train)
ss1 = Scale.transform(X_test)
from sklearn.svm import LinearSVC
model = LinearSVC()
model.fit(ss, y_train)
LSVC_y_pred = model.predict(ss1)
LSVC_y_pred
from sklearn.metrics import accuracy_score, confusion_matrix,
                            classification_report
accuracy_score(y_test, LSVC_y_pred)
```

final prog

```
rbf_svc = svm.SVC(kernel='rbf', C=1.0)
rbf_svc.fit(ss, y_train)
rbfsvc_y_pred = rbf_svc.predict(ss1)
sigmoid_svc = svm.SVC(kernel='sigmoid', C=1.0)
sigmoid_svc.fit(ss, y_train)
sigsvc_y_pred = sigmoid_svc.predict(ss1)
print("Linear SVC :", accuracy_score(y_test, lsvc_ypred))
print("poly SVC :", accuracy_score(y_test, psvc_y_pred))
print("rbf svc:", accuracy_score(y_test, rbfsvc_y_pred))
print("Sigmoid svc:", accuracy_score(y_test, sigsvc_y_pred))

Models = []
models.append(('svc', LinearSVC()))
modelsss.append(('svc_poly', svm.SVC(kernel='poly',
                                     C=1.0)))
modelsss.append(('svc_rbf', svm.SVC(kernel='rbf',
                                    C=1.0)))
modelsss.append(('svc_poly', svm.SVC(kernel='sigmoid',
                                     C=1.0)))

results = []
names = []
scoring = 'accuracy'
for name, model in modelsss:
    kfold = model_selection.KFold(n_split=10)
    cv_results = model_selection.cross_val_score(model,
    results.append(cv_results)
    names.append(cv_results)
    names.append(name)
    print(msg)
```

NOTES

```
Confusion_matrix(y_test, Lsvc_y_pred)

print('classification_report(y_test, Lsvc_y_pred))

from sklearn.linear_model import LogisticRegression

    lgr = LogisticRegression()
    lgr.fit(ss, y_train)
    lr_y_pred = lgr.predict(ss1)
  from .sklearn.svm import linear SVC.
      l_svc = linear SVC()
      l_svc.fit(ss, Y_train)
      lsvc_y_pred = l_svc.predict(ss1)
    print("Logistic Regression :", accuracy_score(y_test, lr_y_pred))
    print("Linear SVC :", accuracy_score(Y_test, lsvc_y_pred))
  from sklearn import svm, model_selection
    model = []
    model.append(('svc', LinearSVC()))
    model.append(('LR', LogisticRegression()))
      results = []
        names = []
        scoring = 'accuracy'
    for name, model in model:
        kfold = model_selection.kfold(n_splits=10)
```

```
cv_results = model_selection.cross_val_score(model, X_train, Y_train,
                                             scoring)
results.append(cv_results)
names.append(name)
msg = "%s: %f (%f)" % (name, cv_results.mean(),
                       cv_results.std())
print(msg)
fig = plt.figure()
fig.suptitle('Comparison between different MLAs')
ax = fig.add_subplot(111)
plt.boxplot(results)
plt.show()
from sklearn.linear_model import SGDClassifier
models = []
models.append(('SVC', LinearSVC()))
models.append(('SGD', SGDClassifier()))
results = []
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train,
                                                 Y_train, cv=kfold, scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(),
                           cv_results.std())
    print(msg)
```

NOTES

```
fig = plt.figure()
fig.suptitle('Comparison between different MLAs')
ax = fig.add_subplot(111)
plt.boxplot(results)
plt.show()
from sklearn.linear_model import SGDClassifier
sgd = SGDclassifier()
sgd.fit(ss, y_train)
sgdc_y_pred = sgd.predict(ss1)
from sklearn.svm import linearSVC
l_svc = LinearSVC()
l_svc.fit(ss, y_train)
l_svc_y_pred = l_svc.predict(ss1)
print("SGDclassifiers", accuracy_score(y_test, sgdc_y_pred))
print("LinearSVC:", accuracy_score(y_test, lsvc_y_pred))
```

3)

```
from sklearn.svm import SVC
l_svc = LinearSVC()
l_svc.fit(ss, y_train)
l_svc_y_pred = l_svc.predict(ss1)
poly_svc = svm.SVC(kernel='poly', C=1.0)
poly_svc.fit(ss, y_train)
psvc_y_pred = poly_svc.predict(ss1)
```