Import Required Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.feature_selection import chi2
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
import pickle
```

RFE_FEATURE FUNCTION

```python
def rfeFeature(indep_X, dep_Y, n):
#creating a list to do RFE for all the models
   rfelist=[]
   log_model = LogisticRegression(solver='lbfgs',max_iter=1000)
   RF = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=0)
   DT = DecisionTreeClassifier(criterion='gini', max_features='sqrt',splitter='best',random_state=0)
   svc_model=SVC(kernel='linear', random_state=0)
   rfemodellist=[log_model,RF,DT,svc_model]
#using for loop, determining feature using RFE for the required models
   for i in rfemodellist:
       print(i)
       log_rfe = RFE(estimator=i, n_features_to_select=n)
       log_fit=log_rfe.fit(indep_X, dep_Y)
       rfelist.append(log_rfe.support_)
   return rfelist
```

- Defining the function rfeFeature by passing input, output and number of features
- Creating an empty list named rfelist
- Defining the models(RF, logistic Regression, SVC, DT) with required parameters and having them in the rfemodellist
- For loop is used to iterate over the models in the rfemodelist and pass them one by one to RFE for feature selection
- Defining log_rfe by passing required estimators and n value to RFE()
- Here estimators are decided with respect to the model and n value is given in the rfeFeature function
- Each model available in the rfemodellist is fitted using fit() by passing the input and output samples from the dataset
- Once the fit is successful append the features selected for each model to rfelist

Split_scalar function

```python
# Function to split test and train from the dataset and to preprocess the input using standard scaler
def split_scalar(indep_X,dep_Y):
    X_train,X_test,Y_train,Y_test=train_test_split(indep_X, dep_Y, test_size=0.25, random_state=0)
    # Feature scaling
    from sklearn.preprocessing import StandardScaler
    sc=StandardScaler()
    X_train=sc.fit_transform(X_train)
    X_test=sc.transform(X_test)
    return X_train,X_test,Y_train,Y_test
```

- The split_scalar function is defined by passing input and output samples
- Training data and test data is splitted using test_train_split function by passing input, output and test_size
- Input data(X_train,X_test) is preprocessed using StandardScaler()
- Splitted input(X_train,X_test) and output(Y_train,Y_test) data is returned back to the function

Cm_prediction Function

```python
# Function for Confusion matrix and accuracy
def cm_prediction(classifier,X_test):
    y_pred=classifier.predict(X_test)
    from sklearn.metrics import confusion_matrix
    cm=confusion_matrix(Y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    Accuracy=accuracy_score(Y_test,y_pred)
    report=classification_report(Y_test,y_pred)
    return classifier,Accuracy,report,X_test,Y_test,cm
```

- The cm_prediction function is defined by passing classifier and test data(X_test)
- Output is predicted using predict() by passing the test data(X_test) and the result is stored in the y_pred variable
- import confusion_matrix from the metrics of the sklearn module
- Confusion matrix is determined using confusion_matrix() by passing actual and predicted output data(Y_test, y_pred) and it is assigned it to cm variable
- import accuracy_score, classification_report from metrics of the sklearn module
- Accuracy is determined using accuracy_score() by passing actual and predicted output data and it is assigned to the Accuracy variable
- classification report is determined using classification_report() by passing actual and predicted output data and it is assigned to the report variable
- Calculated metrics and test data were returned to the function

Function for different models

```python
# Function for Logistic model
def logistic(X_train,Y_train,X_test):
    from sklearn.linear_model import LogisticRegression
    classifier=LogisticRegression(random_state=0)
    classifier.fit(X_train,Y_train)
    classifier,Accuracy, report,X_test,Y_test,cm=cm_prediction(classifier,X_test)
    return classifier,Accuracy,report,X_test,Y_test,cm

# Funcrion for svm linear model
def svm_linear(X_train,Y_train,X_test):
    from sklearn.svm import SVC
    classifier=SVC(kernel='linear', random_state=0)
    classifier.fit(X_train,Y_train)
    classifier,Accuracy,report,X_test,Y_test,cm=cm_prediction(classifier,X_test)
    return classifier,Accuracy,report,X_test,Y_test,cm

# Function for svm non linear model

def svm_NL(X_train,Y_train,X_test):
    from sklearn.svm import SVC
    classifier=SVC(kernel='rbf', random_state=0)
    classifier.fit(X_train,Y_train)
    classifier, Accuracy, report, X_test, Y_test, cm = cm_prediction(classifier,X_test)
    return classifier, Accuracy, report, X_test, Y_test, cm
```

- Different functions are created for differen models like Logistic Regression, SVM Random Forest, Decision tree.
- Import required model functions from the sklearn module.
- Classifier is defined using model function like(logistic Regression(), SVC() by passing required parameters
- Model is created using fit() by passing training data(X_train,Y_train)
- Metrics for models are calculated by calling the cm_prediction() and the result is stored in the required metric variables
- Metrics and input test data is returned to model function
- All the above steps are same for each model

RFE_classification Function

```
# Function for preparing table for the RFE with best accuracy for the models
def RFE_classification(acclog, accsvml, accsvmnl, accknn, accnav, accdes, accrf):
    rfedataframe=pd.DataFrame(index=['Logistic','SVC','Random','DecisionTree'],columns=['Logistic', 'SVML','SVMnl','KNN','Naive',
                                                                                         'Decision','Random'])
    for number,idex in enumerate(rfedataframe.index):
        rfedataframe['Logistic'][idex]=acclog[number]
        rfedataframe['SVML'][idex]=accsvml[number]
        rfedataframe['SVMnl'][idex]=accsvmnl[number]
        rfedataframe['KNN'][idex]=accknn[number]
        rfedataframe['Naive'][idex]=accnav[number]
        rfedataframe['Decision'][idex]=accdes[number]
        rfedataframe['Random'][idex]=accrf[number]
    return rfedataframe
```

- RFE_classification function is defined by passing accuracy of every model created
- rfedataframe is created using pandas having rows with respect to rfefeature() and columns with respect to models created using different functions
- For loop with enumerate() is used to store accuracy value of each row and column i.e accuracy of each model using rfefeature
- Overall dataframe is returned back to function as it contains accuracy values for each model

Defining dataset

```
dataset1=pd.read_csv("prep.csv",index_col=None)

df2=dataset1

df2=pd.get_dummies(df2, drop_first=True)

indep_X=df2.drop('classification_yes',axis=1)

dep_Y=df2['classification_yes']
```

- Read the csv file using read_csv of pandas and assign the data to dataset1
- Making a copy of the dataset1 and storing it to df2
- Converting the categorical to numerical values using get_dummies() by passing df2 and assign the converted data to df2
- Dropping the label column(classification_yes) from df2 using drop() and store it to variable indep_X. Here axis=1 indicates that the operation is to be performed on columns
- Label column(classification_yest) is stored to variable dep_Y

```
rfelist=rfeFeature(indep_X,dep_Y,3)
```

```
acclog=[]
accsvml=[]
accsvmnl=[]
accknn=[]
accnav=[]
accdes=[]
accrf=[]
```

```
for i in rfelist:
    X_selected=indep_X.iloc[:,i]
    X_train,X_test,Y_train,Y_test=split_scalar(X_selected,dep_Y)
    classifier,Accuracy,report,X_test,Y_test,cm =logistic(X_train,Y_train,X_test)
    acclog.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = svm_linear(X_train,Y_train,X_test)
    accsvml.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = svm_NL(X_train,Y_train,X_test)
    accsvmnl.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = Naive(X_train,Y_train,X_test)
    accnav.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = knn(X_train,Y_train,X_test)
    accknn.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = Decision(X_train,Y_train,X_test)
    accdes.append(Accuracy)
    classifier,Accuracy,report,X_test,Y_test,cm = random(X_train,Y_train,X_test)
    accrf.append(Accuracy)

result = RFE_classification(acclog,accsvml,accsvmnl,accknn,accnav,accdes,accrf)
```

- Recursive Feature elimination is applied for each model(this is the models stated in rfeFeature function) by calling rfeFeature() by passing input, output and number of features
- create an empty list for accuracy of each model
- For loop is used to iterate models one by one from rfelist and apply the rfefeature of them to every model function by passing test and training data
- Accuracy for each model by applying rfefeature is calculated and stored to respective list(acclog,accsvml,accsvml..etc)
- Accuracy for each row and column is updated by calling RFE_classification function and the result is stored to result variable

Output (k:3,4,5,6)

```
[6]: result
     #3
```

[6]:

	Logistic	SVML	SVMnl	KNN	Naive	Decision	Random
Logistic	0.94	0.94	0.94	0.94	0.94	0.94	0.94
SVC	0.91	0.92	0.93	0.93	0.86	0.91	0.94
Random	0.93	0.93	0.94	0.95	0.74	0.95	0.97
DecisionTree	0.87	0.87	0.87	0.87	0.87	0.87	0.87

```
result
#4
```

	Logistic	SVML	SVMnl	KNN	Naive	Decision	Random
Logistic	0.95	0.95	0.95	0.95	0.95	0.95	0.95
SVC	0.93	0.93	0.94	0.93	0.91	0.91	0.94
Random	0.97	0.97	0.97	0.96	0.84	0.96	0.96
DecisionTree	0.96	0.96	0.96	0.96	0.96	0.96	0.96

```
result
#5
```

	Logistic	SVML	SVMnl	KNN	Naive	Decision	Random
Logistic	0.98	0.98	0.98	0.98	0.98	0.98	0.98
SVC	0.97	0.97	0.97	0.96	0.87	0.93	0.97
Random	0.97	0.98	0.98	0.98	0.91	0.96	0.98
DecisionTree	0.99	0.99	0.99	0.99	0.99	0.99	0.99

```
result
#6
```

	Logistic	SVML	SVMnl	KNN	Naive	Decision	Random
Logistic	0.98	0.98	0.98	0.98	0.98	0.99	0.98
SVC	0.98	0.98	0.99	0.97	0.93	0.96	0.96
Random	0.97	0.99	0.98	0.98	0.91	0.97	1.0
DecisionTree	0.99	0.99	0.99	0.99	0.99	0.99	0.99