```
# Import important library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
%matplotlib inline

import numpy as np
import pylab as pl
import pandas as pd
import matplotlib.pyplot as plt

from scipy import interp
from itertools import cycle

from sklearn import svm
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import cross_val_score,StratifiedKFold
from sklearn.preprocessing import OneHotEncoder, label_binarize, StandardScaler, MinMaxScaler
from sklearn.metrics import classification_report,confusion_matrix, roc_curve, auc
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import ExtraTreesClassifier, RandomForestClassifier, GradientBoostingClassi
from sklearn.feature_selection import RFE, SelectKBest, chi2, SelectFromModel
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import ExtraTreesClassifier
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: DeprecationWarnin
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

Read the input file and check the data dimension

```
from google.colab import drive
drive.mount('/content/drive')
```

```
data = pd.read_csv('/content/drive/My Drive/8sept/assignment/german_credit.csv')
```

```
data
```

```
# You can access from https://www.kaggle.com/uciml/german-credit
#Read input file and understand the data
# "default" is my dependent variable
```

```
(1000, 21)
Index(['default', 'account_check_status', 'duration_in_month',
       'credit_history', 'purpose', 'credit_amount', 'savings',
       'present_emp_since', 'installment_as_income_perc',
       'personal_status_sex', 'other_debtors', 'present_res_since', 'property',
       'age', 'other_installment_plans', 'housing', 'credits_this_bank', 'job',
       'people_under_maintenance', 'telephone', 'foreign_worker'],
      dtype='object')
```

	default	account_check_status	duration_in_month	credit_history	purpose	cr
0	0	< 0 DM	6	critical account/ other credits existing (not ...	domestic appliances	
1	1	0 <= ... < 200 DM	48	existing credits paid back duly till now	domestic appliances	
2	0	no checking account	12	critical account/ other credits existing (not ...	(vacation - does not exist?)	
3	0	< 0 DM	42	existing credits paid back duly till now	radio/television	
4	1	< 0 DM	24	delay in paying off in the past	car (new)	

5 rows × 21 columns

Q1 Randomly select 50% data for this use case(1 Marks)

Hint: Use train_test_split

```
# Lets build a Ensemble model but need to modify the dataset first
```

Q2.Prepare the model data by converting non-numeric to dummy (1 Marks)

Hint: Use get_dummies

```
data_account_check_status = pd.get_dummies(data['account_check_status'])
data_savings = pd.get_dummies(data['savings'])
```

```
data_present_emp_since = pd.get_dummies(data['present_emp_since'])

data_credit_history = pd.get_dummies(data['credit_history'])
data_personal_status_sex = pd.get_dummies(data['personal_status_sex'])
data_property = pd.get_dummies(data['property'])
```

```
data_concat = pd.concat([data, data_account_check_status, data_savings,  data_present_emp_since ,
print (data_concat.head())
```

```
data_concat.drop(['account_check_status', 'savings','present_emp_since' , 'credit_history','perso
print (data_concat.head())
```

```
data_concat.drop(['other_debtors', ], inplace=True, axis=1)
```

```
data_concat
```

```
# Print Shape of model data
```

```
(500, 61)
```

Check for highly correlated variables but don't required any treatment for this use case

```
data_concat.corr().transpose()
```

Drop the original variables which are converted to dummy

```
data_concat.drop(['account_check_status', 'savings','present_emp_since' , 'credit_history','persc
```

```
sns.pairplot(data_concat)
```

Q3 Split Train/Test data 70:30 ratio(1 Marks)

Hint:from sklearn.model_selection import train_test_split

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(data_concat.drop(['< 0 DM' ,'no checking account'], axis=1
```

```
train_set.describe()
```

```
test_set.describe()
```

Q4 Build Random Forest Model(1 Marks)

Hint:from sklearn.ensemble import RandomForestClassifier using n_jobs=2,n_estimators=500,criterion="entro

```
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
train_labels = train_set.pop("default")
test_labels = test_set.pop("default")
```

```
randomforest_model = RandomForestClassifier(n_jobs=2,n_estimators=500,criterion="entropy", random
randomforest_model.fit(train_set, train_labels)
```

```
Importance = pd.DataFrame({'Importance':randomforest_model.feature_importances_*100}, index=train
Importance.sort_values('Importance', axis=0, ascending=True).plot(kind='barh', color='r', )
```

Q5 Calculate Confusion Matrix and Accuracy score (1 Marks)

Hint: Use confusion_matrix and accuracy_score

```
import numpy as np
import pylab as pl
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(style="ticks", color_codes=True)
import pydotplus

from IPython.display import Image
from collections import defaultdict
from sklearn.dummy import DummyClassifier
from sklearn import svm, tree
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, label_binarize, StandardScaler, Mi
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report,confusion_matrix, roc_curve, auc
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import ExtraTreesClassifier, RandomForestClassifier, GradientBoostingClassi
from sklearn.feature_selection import RFE, SelectKBest, chi2, SelectFromModel
```

```
from sklearn.linear_model import LogisticRegression, SGDClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report,confusion_matrix, roc_curve, auc
```

Q6 Show the list of the features importance(1 Marks)

1 cell hidden

Q7 K-fold cross-validation(2 Marks)

k-fold cross validation(without stratification)

Usually k is set as 10-20 in practical settings, depends on data set size

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


# Use below values
num_folds = 10
seed = 77


#Validate the Random Forest model build above using k fold


def get_eval(clf, X_train, y_train,y_test,y_pred):
    # Cross Validation to test and anticipate overfitting problem
    scores1 = cross_val_score(clf, X_train, y_train, cv=10, scoring='accuracy')
    scores2 = cross_val_score(clf, X_train, y_train, cv=10, scoring='precision')
    scores3 = cross_val_score(clf, X_train, y_train, cv=10, scoring='roc_auc')
    # The mean score and standard deviation of the score estimate
    print("Cross Validation Accuracy: %0.2f (+/- %0.2f)" % (scores1.mean(), scores1.std()))
    print("Cross Validation Precision: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std()))
    print("Cross Validation roc_auc: %0.2f (+/- %0.2f)" % (scores3.mean(), scores3.std()))
    # Create and print confusion matrix
    abclf_cm = confusion_matrix(y_test,y_pred)
    print(abclf_cm)
    return




# Calculate score standard deviation using std()
```

Q8 Print the confusion matrix(1 Marks)

Q9.Classification accuracy:

percentage of correct predictions and Calculate sensitivity (or True Positive Rate or Recall) and Precision. (1 Ma

Q10.Plot Receiver Operating Characteristic (ROC) Curv

```
#Hint: Use roc_curve
def get_roc (y_test,y_pred):
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, _ = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    #Plot of a ROC curve
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.0])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
    return
```

ROC curve can help you to choose a threshold that balances sensitivity and specificity in a way that makes sens

Q11. Calculate AUC(the percentage of the ROC plot tha - optional

Bootstrapping (Bonus)

Given a dataset of size n, a bootstrap sample is created by sampling n instances uniformly from the data (with/without replacement)

Create a model with each bootstrap sample and validate it with the test set

Final result is calculated by averaging the accuracy of models

4 cells hidden