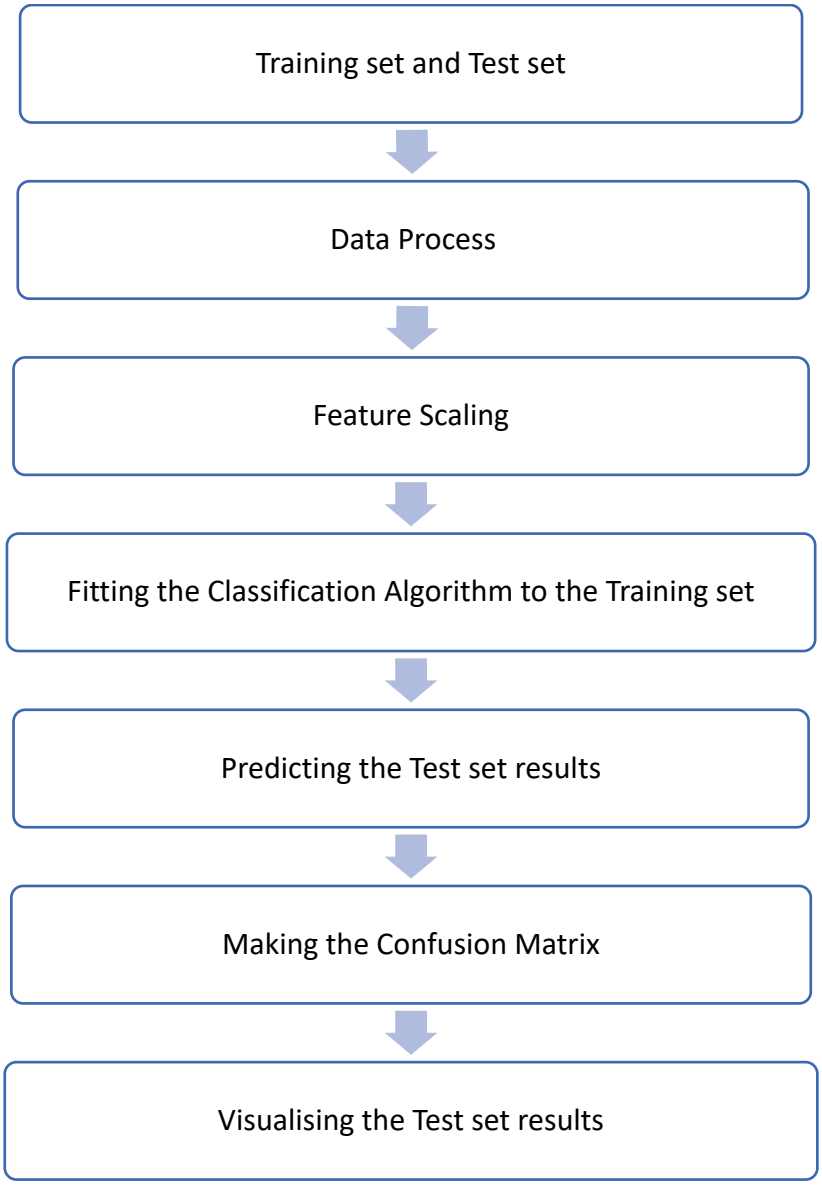
MIDTERM REPORT

Advance in Data Sciences and Architecture
INFO 7390 - SPRING 2017

PROFESSOR:
SRIKANTH KRISHNAMURTHY

TEAM MEMBERS (Team 10):
AASHRI TANDON
PRAGATI SHAW
SARTHAK AGARWAL


Training set and Test set
Data Process
Feature Scaling
Fitting the Classification Algorithm to the Training set
Predicting the Test set results
Making the Confusion Matrix
Visualising the Test set results

1. Programmaticallydownloads Q12005 and Q22005origination data and pre-processesit.

```
############### Create Session ###############
USERNAME = 'wangying0327@hotmail.com'
PASSWORD = 'Y3ZUK;\D'

payload = {
    "username": USERNAME,
    "password": PASSWORD
}

session_requests = requests.session()

login_url = "https://freddiemac.embs.com/FLoan/secure/auth.php"

result = session_requests.post(
    login_url,
    data=payload,
    headers=dict(referer=login_url)
)

url = 'https://freddiemac.embs.com/FLoan/Data/download.php'
agreement_payload = {
    "accept": "Yes",
    "action": "acceptTandC",
    "acceptSubmit": "Continue"
}
result = session_requests.post(
    url,
    agreement_payload,
    headers=dict(referer=url)
)
```

2. Builds a Logistic regression model for the CURRENTLOANDELINQUENCYSTATUS

```
'''--------------------------Logistic Regression-----------------------------------------------------'''
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

# Fitting the Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs')
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

3. using Q12005 data as training data (col 4). Note anytime col 4 is > 0, add a new variable as Delinquent. Use this variable as your “Y” variable. IGNORE COL 4 AND DON’T USE IT IN YOUR MODEL.

```
def f(row):
    if row['current_loan_delinquency_status'] > 0:
        val = 1
    else:
        val = 0
    return val
# Create dummy variables
df1_dummies = pd.get_dummies(df1[['repurchase_flag', 'modification_flag']])
df2_dummies = pd.get_dummies(df2[['repurchase_flag', 'modification_flag']])

df1_d = df1.drop(['loan_sequence_no', 'repurchase_flag', 'modification_flag'], axis=1)
df2_d = df2.drop(['loan_sequence_no', 'repurchase_flag', 'modification_flag'], axis=1)

global df1_final
global df2_final
df1_final = pd.concat([df1_d, df1_dummies], axis=1)
df2_final = pd.concat([df2_d, df2_dummies], axis=1)

# create target variable
df1_final['Deliquent'] = df1_final.apply(f, axis=1)
df2_final['Deliquent'] = df2_final.apply(f, axis=1)

# Create training and testing set

X_train = df1_final.drop(['current_loan_delinquency_status', 'Deliquent'], axis=1)
y_train = df1_final['Deliquent']

X_test = df2_final.drop(['current_loan_delinquency_status', 'Deliquent'], axis=1)
y_test = df2_final['Deliquent']

X_train = preprocessing.minmax_scale(np.array(X_train).astype(float))  # scale between 0 and 1
X_test = preprocessing.minmax_scale(np.array(X_test).astype(float))

return X_train, y_train, X_test, y_test
```

4. ValidatesagainstQ22005 data and selects the best Classification model

   Random Forest

```
# random Forest
def rf(X_train, y_train, X_test, y_test, return_dict_rf):
    rf = RandomForestClassifier(n_estimators=100)
    rf.fit(X_train, y_train)

    y_train_predicted = rf.predict(X_train)
    y_test_predicted = rf.predict(X_test)

    conf_mat_rf = metrics.confusion_matrix(y_test, y_test_predicted)
    print(conf_mat_rf)

    # Compute ROC curve and AUC (Area under the Curve
    false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, y_test_predicted)

    roc_auc_rf = metrics.auc(false_positive_rate, true_positive_rate)
    return_dict_rf['roc_auc_rf'] = roc_auc_rf
    return_dict_rf['conf_mat_rf'] = conf_mat_rf

    # Plot ROC Curve
    plt.title("Random Forest")
    plt.plot(false_positive_rate, true_positive_rate, 'b',
             label='AUC = %0.2f' % roc_auc_rf)
    plt.legend(loc='lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([-0.1, 1.2])
    plt.ylim([-0.1, 1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

   Neural Network

```
def nn(X_train, y_train, X_test, y_test, return_dict_nn):

    nn = MLPClassifier()

    nn.fit(X_train, y_train)

    y_train_predicted = nn.predict(X_train)
    y_test_predicted = nn.predict(X_test)

    conf_mat_nn = metrics.confusion_matrix(y_test, y_test_predicted)
    print(conf_mat_nn)

    # Compute ROC curve and AUC (Area under the Curve
    false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, y_test_predicted)

    roc_auc_nn = metrics.auc(false_positive_rate, true_positive_rate)
    return_dict_nn['roc_auc_nn'] = roc_auc_nn
    return_dict_nn['conf_mat_nn'] = conf_mat_nn

    # Plot ROC Curve
    plt.title("Neural Network")
    plt.plot(false_positive_rate, true_positive_rate, 'b',
             label='AUC = %0.2f' % roc_auc_nn)
    plt.legend(loc='lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([-0.1, 1.2])
    plt.ylim([-0.1, 1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

5. H20

```
: import h2o

h2o.init(strict_version_check=False,port=9957)
```

```
Checking whether there is an H2O instance running at http://localhost:9957..... not found.
Attempting to start a local H2O server...
  Java Version: openjdk version "1.8.0_121"; OpenJDK Runtime Environment (Zulu 8.20.0.5-macosx) (build 1.8.0_121-b1
5); OpenJDK 64-Bit Server VM (Zulu 8.20.0.5-macosx) (build 25.121-b15, mixed mode)
  Starting server from /anaconda3/lib/python3.6/site-packages/h2o/backend/bin/h2o.jar
  Ice root: /var/folders/hf/c05h_6kd5wlds2ysv1yvpy1c0000gn/T/tmpdo3zi7w5
  JVM stdout: /var/folders/hf/c05h_6kd5wlds2ysv1yvpy1c0000gn/T/tmpdo3zi7w5/h2o_wangying_started_from_python.out
  JVM stderr: /var/folders/hf/c05h_6kd5wlds2ysv1yvpy1c0000gn/T/tmpdo3zi7w5/h2o_wangying_started_from_python.err
  Server is running at http://127.0.0.1:9957
Connecting to H2O server at http://127.0.0.1:9957... successful.
```

H2O cluster uptime:	01 secs
H2O cluster timezone:	America/New_York
H2O data parsing timezone:	UTC
H2O cluster version:	3.22.0.2
H2O cluster version age:	15 days
H2O cluster name:	H2O_from_python_wangying_kbgw7l
H2O cluster total nodes:	1
H2O cluster free memory:	3.556 Gb
H2O cluster total cores:	8
H2O cluster allowed cores:	8
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:9957
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	XGBoost, Algos, AutoML, Core V3, Core V4
Python version:	3.6.6 final

Out[27]:

	model_id	auc	logloss	mean_per_class_error	rmse	mse
0	StackedEnsemble_AllModels_AutoML_20181207_133501	0.911731	0.139201	0.210976	0.187376	0.035110
1	StackedEnsemble_BestOfFamily_AutoML_20181207_1...	0.911731	0.139201	0.210976	0.187376	0.035110
2	DRF_1_AutoML_20181207_133501	0.907181	0.155591	0.219043	0.203407	0.041374
3	XRT_1_AutoML_20181207_133501	0.906353	0.157899	0.215183	0.205562	0.042256
4	GLM_grid_1_AutoML_20181207_133501_model_1	0.745118	0.221634	0.329522	0.241570	0.058356

In [28]:
```
model_set=aml_leaderboard_df['model_id']
mod_best=h2o.get_model(model_set[0])
```

Best Model:
StackedEnsemble_AllModels_AutoML_20181207_133501

6. Parameterize the input (example it should take Q12005) and modify the code so that it outputs the 5 parameters listed in the matrix below.

matrix_classification

	Quarter	No_of_actual_delq	No_of_pred_delq	No_of_records	No_of_delq_properly_classified	No_of_nonDelq_improperly_classified_as_delq
0	Q12005	8500	3761	200000	741	3020
0	Q12005	8500	3761	200000	741	3020

7. Write another script that calls the above classification script from Q11999-Q42016 and computes the following matrix.

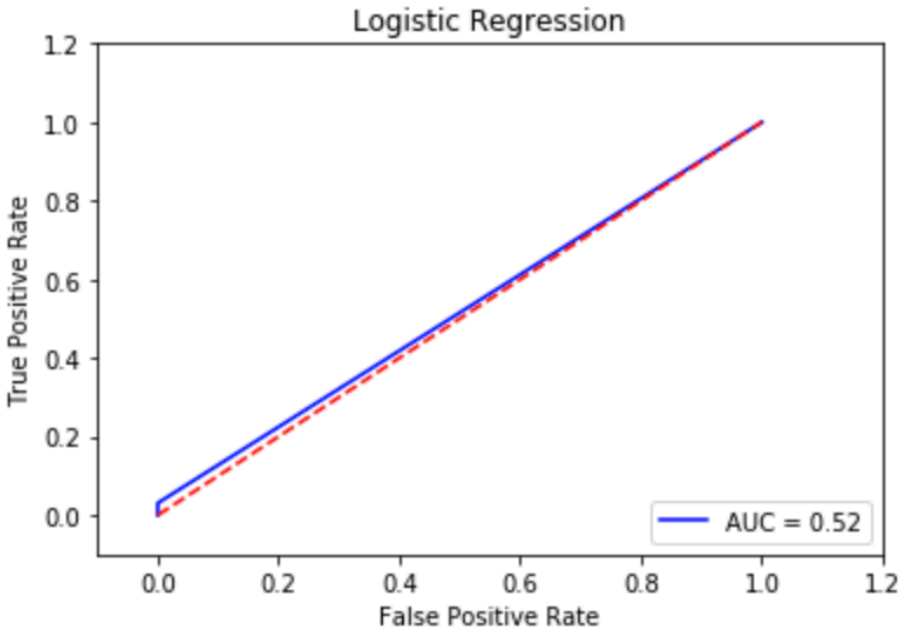
```
arg_len = len(sys.argv)
quarters = []
end = ''
if arg_len == 3:
    startQuarter = sys.argv[1]
    endQuarter = sys.argv[2]
    end = endQuarter
    while(startQuarter != endQuarter):
        print(startQuarter)
        quarters.append(startQuarter)
        startQuarter = get_next_quarter(startQuarter)
    quarters.append(endQuarter)
elif arg_len == 2:
    startQuarter = sys.argv[1]
    print("---" + startQuarter)
    quarters.append(startQuarter)
    end = startQuarter
else:
    print("running for default Q12005")
    quarter = 'Q12005'
    quarters.append(quarter)
    end = quarter
download
```

8. • Extra credit: Note that each invocation of is independent. There is scope to parallelize it. Parallelize this so that you can run two or more models concurrently (task parallel). Review R and Python packages that can be used to parallelize it to generate the below matrix as a dataframe and export it to a csvfile.

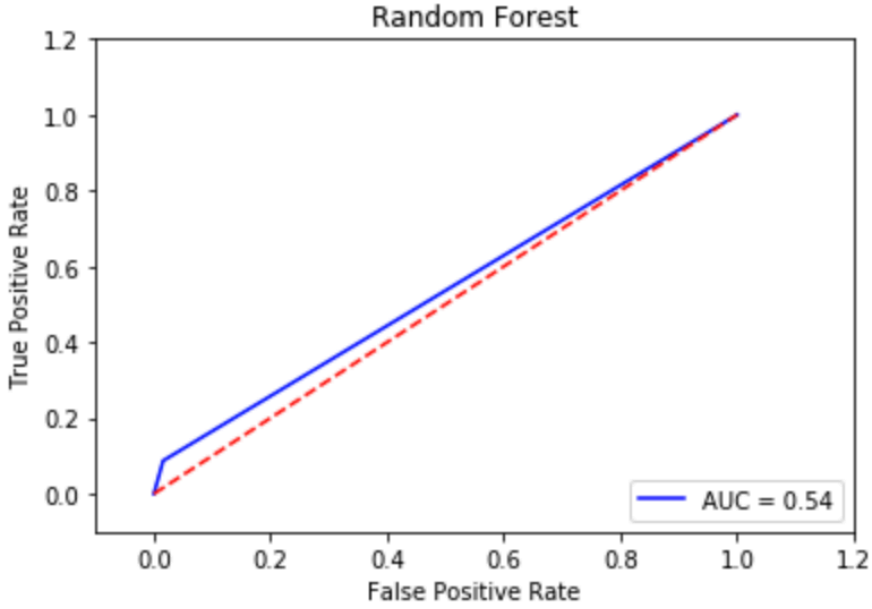
```
for q in quarters:
    nextQuarter = get_next_quarter(q)
    load_data_df(q, nextQuarter)
    remove_nan(df1)
    remove_nan(df2)
    X_train, y_train, X_test, y_test = process_data()
    print("process data executed")

    p1 = Process(target=logred,args=(X_train, y_train,X_test,y_test,return_dict_logred))
    p2 = Process(target=rf,args=(X_train, y_train,X_test,y_test,return_dict_rf))
    p3 = Process(target=nn,args=(X_train, y_train,X_test,y_test,return_dict_nn))
    p1.start()
    p2.start()
    p3.start()
    p1.join()
    p2.join()
    p3.join()
    matrix=genMatrix(q,return_dict_logred['roc_auc_logred'],return_dict_rf['roc_auc_rf'],return_dict_nn['
        roc_auc_nn'],return_dict_logred['conf_mat_logred'],return_dict_rf['conf_mat_rf'],return_dict_nn['
        conf_mat_nn'],matrix)

print(matrix)
matrix.to_csv('matrix_classification.csv')
```

Graph：

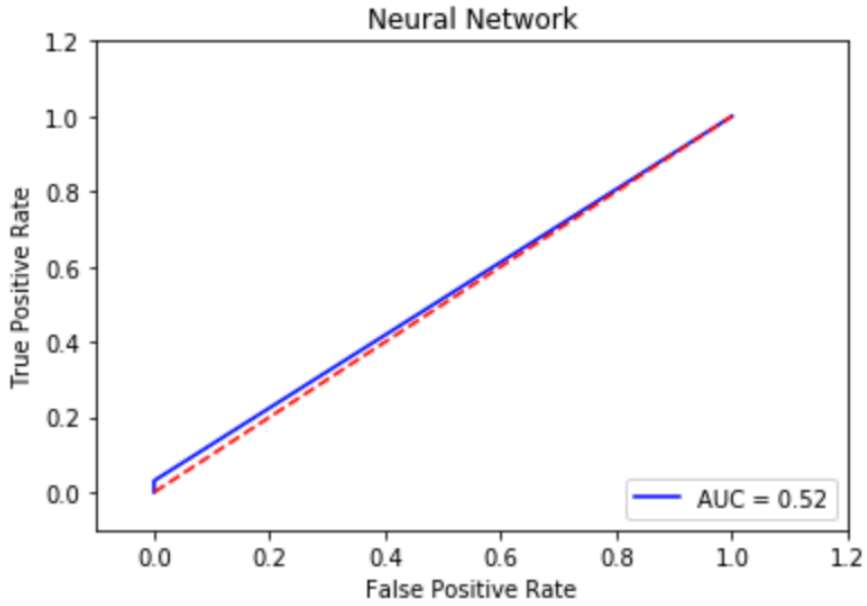
```
[[191498      2]
 [  8239    261]]
```




```
[[188480   3020]
 [  7759    741]]
```



```
[[191498      2]
 [  8239    261]]
```



	Quarter	No_of_actual_delq	No_of_pred_delq	No_of_records	\
0	Q12005	8500.0	3761.0	200000.0	
0	Q12005	8500.0	3761.0	200000.0	

	No_of_delq_properly_classified	No_of_nonDelq_improperly_classified_as_delq
0	741.0	3020.0
0	741.0	3020.0

]: