Unit 10: Final Project - Demonstration and Instructions- Python

1 – Project Template

2 – Example - Classification

3 – Project Orientation

1 - Project Template

- Introduction
- Using Python - Jupyter
- Step-by-Step

1 - Project Template

Introduction

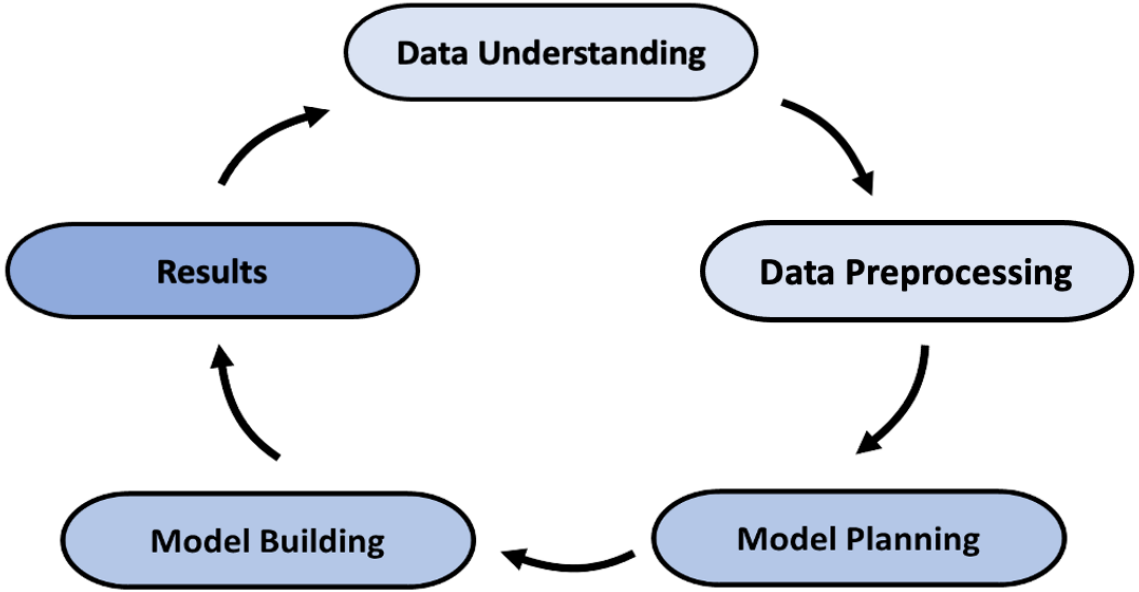
- *Create machine learning predictive models end-to-end is critically important - starting from the dataset to obtaining the final model*

- *We will put together all the materials previously discussed to create the predictive models*

- *As a base, we will use the five phases of a machine learning lifecycle project*

1 - Project Template

Introduction



Notes

- *As we have previously seen tasks will be divided in each phase*
- *Lets put all together in Jupyter Notebook*

1 - Project Template

Using Python - Jupyter

```
# Project Template- Jupyter
#Introduction
    # Define the Problem
    # Load libraries
    # Load dataset

# Phase I - Data Understanding
    # Understand data with statistics
    # Understand data with visualizations

# Phase II - Data Preprocessing
    # Feature Selection
    # Data Transform

# Phase III - Model Planning
    # Split-out validation dataset
    # Performance evaluation
    # Performance metrics

# Phase IV - Model Building
    # Training and Comparing Models
    # Algorithm Tuning

# Phase V - Results
    # Finalize Model and predict on validation dataset
    # Save model
```

1 - Project Template

Using Python - Jupyter

- *How to proceed in Jupyter:*
  - *Create a new notebook in Jupyter using Python 3*
  - *Copy the project template from the previous slide, and*
  - *Paste it into the empty notebook*
  - *Start to fill it in*

1 - Project Template

Step-by-Step

Introduction

a) Load Python modules, classes and functions

b) Load dataset from CSV

1 - Project Template

Step-by-Step

Data Understanding

Phase I - Data Understanding

a) Use descriptive statistics such as summaries.

b) Use data visualizations such as plots with Matplotlib

1 - Project Template

Step-by-Step

Data Preprocessing

Phase II - Data Preprocessing

a) Clean data by removing duplicates, marking missing values and imputing missing values

b) Select best features selection where redundant features may be removed, and new features developed

c) Transform data where attributes are scaled or redistributed in order to best expose the structure of the problem later to learning algorithms

1 - Project Template

Step-by-Step

Model Planning

Phase III - Model planning

a) Separate out a validation dataset to use for later confirmation of the skill of developed model.

b) Define test options using scikit-learn such as cross-validation, and

c) Define the metrics to use for model evaluations

1 - Project Template

Model Building

Step-by-Step

Phase IV - Model Building

a) Spot-check a suite of linear and nonlinear machine learning algorithms and comparing the estimated accuracy of algorithms.

b) Improve accuracy by searching for a combination of parameters for each algorithm that yields the best results

c) Use the optimal tuned model to make predictions on unseen data

1 - Project Template

Step-by-Step

Results

Phase V - Results

a) Create a standalone model using the selected algorithm with the tuned parameters to train on the whole dataset.

b) Save this optimal model to a file to use later

Unit 10: Final Project - Demonstration and Instructions- Python

1 – Project Template

2 – Example - Classification

3 – Project Orientation

2 – Example - Classification

- *Introduction*
  - *Define the problem*
  - *Load libraries*
  - *Load dataset*
- *Phase I - Data Understanding*
  - *Understand data with statistics*
  - *Understand data with Visualization*
- *Phase II - Data Preprocessing*
  - *Data transform*
- *Phase III - Model Planning*
  - *Split-out validation dataset*
  - *Performance evaluation*
  - *Performance metrics*
- *Phase IV - Model Building*
  - *Training and comparing models*
  - *Algorithm tunning*
- *Phase V - Results*
  - *Finalize model and predict on validation dataset*
  - *Save model*

2 – Example - Classification

Introduction - Define the problem

- *Antiretroviral therapy (ART) has greatly increased life expectancy for people living with HIV (HIV+), but cardiovascular disease (CVD) is now a leading health threat in patients receiving ART.*
- *HIV infection confers increased risk for CVD, and is linked to ART-induced dyslipidemia, the chronic inflammation induced by HIV infection.*
- *As HIV+ individuals' lifespans approach that of the general population, the management of modifiable cardiovascular risk factors becomes increasingly important.*
- *The focus of this example will be the CVD in ART dataset. The problem is to predict which patients are likely to develop CVD after ART treatment.*

2 – Example - Classification

Introduction - Load libraries

- *Load the needed libraries for the project*

```
# Load libraries
from matplotlib import pyplot
import numpy as np
import pandas as pd
from pandas import read_csv
from pandas import to_numeric
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

2 – Example - Classification

Introduction - Load Dataset

- *Download the dataset "CVD_in_ART_HDR.csv" from the online learning platform*
- *Load the dataset in the corresponding variable "dataset"*

```
# Load dataset
dataset = read_csv('CVD_in_ART_HDR.csv')
```

Notes:
At the same time, open the "CVD_in_ART_HDR.csv" and "code_book.txt" files to get familiar with the dataset

2 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Confirm dimensions of the dataset.*

Code:

```
# shape
print(dataset.shape)
```

Output:

```
(153, 12)
```

2 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Data type of each attribute*

Code:

```
# types
set_option('display.max_rows', 500)
print(dataset.dtypes)
```

Output:

```
IncidentDM              int64
IncidentObese           int64
IncidentOvwtobese       int64
Dyslipidemia            int64
Sex                     int64
DMEnd                   int64
DMEndHDL              float64
DurHIVDMendYrs        float64
BslnBMI               float64
BMILast               float64
DurHIVBMIEndYrs       float64
DurARTTChol           float64
DrugStatus              int64
AvgDrinkPerMo           int64
TobaccoUse              int64
Depression              int64
HTN                     int64
Framingham risk       float64
CVD                     int64
dtype: object
```

2 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Verify the 20 first rows of the data.*

Code:

```
# head
set_option('display.width', 100)
print(dataset.head(20))
```

Output:

	IncidentDM	IncidentObese	IncidentOvwtobese	Dyslipidemia	Sex	DMEnd	DMEndHDL	DurHIVDMendYrs	BslnBMI	BMILast	DurHIVBMIEndYrs	DurARTTChol	DrugStatus	AvgDrinkPerMo	TobaccoUse	Depression	HTN	Framingham risk	CVD
0	1	2	2	1	2	2	-99.0	12.38	32.82	35.01	10.99	5.13	1	3	1	1	0	-99.00	1
1	0	1	2	1	2	2	-77.0	-77.00	27.21	27.96	12.95	11.60	1	6	1	0	0	-99.00	1
2	1	2	2	1	2	2	-99.0	7.41	30.14	33.01	5.60	55.80	1	0	1	1	0	0.03	1
3	1	1	1	1	2	2	-99.0	17.60	21.70	22.40	21.09	25.67	1	4	2	1	1	0.05	0
4	0	1	1	1	2	2	-77.0	-77.00	20.48	20.66	8.81	50.73	1	0	1	1	0	0.01	0
5	1	1	1	1	1	2	-99.0	7.16	22.64	22.47	7.58	44.57	1	12	2	0	0	-99.00	0
6	0	1	1	1	1	1	-77.0	-77.00	21.95	23.42	5.65	4.23	1	1	1	1	0	-99.00	0
7	1	1	1	1	2	1	59.0	14.09	23.62	23.98	14.47	31.57	1	1	3	1	0	0.02	0
8	1	1	1	1	2	2	46.0	11.96	24.24	22.87	8.45	8.13	1	0	1	1	0	0.02	1
9	1	2	2	1	2	1	69.0	5.13	30.76	29.38	6.01	44.23	1	0	1	1	0	0.01	0
10	1	1	2	1	2	2	-99.0	7.56	25.33	25.74	7.44	0.00	1	0	1	1	0	-99.00	1
11	1	1	2	1	1	4	-99.0	4.96	26.44	25.75	5.48	12.80	3	0	3	0	0	-99.00	1
12	1	1	1	1	1	2	-99.0	3.45	24.72	25.02	4.68	11.63	1	6	1	1	0	-99.00	1
13	1	1	1	1	1	1	-99.0	4.66	19.77	25.32	5.05	14.17	1	1	1	1	0	-99.00	0
14	0	1	1	0	2	0	-99.0	-88.00	19.32	21.04	8.53	-88.00	1	0	1	1	0	-99.00	0
15	1	1	1	1	1	2	66.0	4.73	22.65	24.97	2.45	45.53	1	2	1	1	0	0.03	1
16	1	1	1	1	1	2	58.0	12.27	22.47	22.92	12.46	33.73	1	2	1	1	0	0.01	0
17	1	1	2	1	2	1	-99.0	9.74	25.63	27.99	9.98	15.13	1	0	2	1	0	-99.00	0
18	1	1	1	1	2	2	50.0	16.58	23.40	24.43	14.25	37.60	1	0	1	1	0	0.04	1
19	1	1	2	1	2	1	-99.0	18.69	28.70	33.37	18.04	26.83	1	0	1	0	0	-99.00	0

2 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Summarize the statistics of each attribute*

Code:

```
# descriptions, change precision to 3 places
set_option('precision', 3)
print(dataset.describe())
```

Output:

	IncidentDM	IncidentObese	IncidentOvwtobese	Dyslipidemia	Sex	DMEnd	DMEndHDL	\
count	153.000	153.000	153.000	153.000	153.000	153.000	153.000	
mean	0.791	0.954	1.216	0.935	1.569	1.386	-34.257	
std	0.408	0.600	0.725	0.248	0.497	0.836	71.954	
min	0.000	0.000	0.000	0.000	1.000	0.000	-99.000	
25%	1.000	1.000	1.000	1.000	1.000	1.000	-99.000	
50%	1.000	1.000	1.000	1.000	2.000	1.000	-88.000	
75%	1.000	1.000	2.000	1.000	2.000	2.000	46.000	
max	1.000	2.000	2.000	1.000	2.000	4.000	83.000	

	DurHIVDMendYrs	BslnBMI	BMILast	DurHIVBMIEndYrs	DurARTTChol	DrugStatus	AvgDrinkPerMo
count	153.000	153.000	153.000	153.000	153.000	153.000	153.000
mean	-15.942	9.464	11.358	-15.728	11.851	1.190	5.948
std	40.626	41.194	40.696	41.735	31.249	0.535	38.935
min	-111.000	-99.000	-99.000	-111.000	-99.000	1.000	0.000
25%	0.110	20.600	21.400	0.140	5.800	1.000	0.000
50%	3.120	23.620	24.590	4.160	13.600	1.000	1.000
75%	8.030	27.560	28.990	8.450	28.770	1.000	2.000
max	23.990	46.000	45.060	21.950	56.170	3.000	480.000

	TobaccoUse	Depression	HTN	Framingham risk	CVD
count	153.000	153.000	153.000	153.000	153.000
mean	1.386	0.824	0.196	-45.915	0.346
std	0.699	0.382	0.398	49.559	0.477
min	1.000	0.000	0.000	-99.000	0.000
25%	1.000	1.000	0.000	-99.000	0.000
50%	1.000	1.000	0.000	0.010	0.000
75%	2.000	1.000	0.000	0.030	1.000
max	3.000	1.000	1.000	0.210	1.000

2 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Breakdown of class values*

Code:

```
# class distribution
print(dataset.groupby('CVD').size())
```

Output:

```
CVD
0    100
1     53
```

2 – Example - Classification

Data Understanding

Phase I - Data Understanding – with Visualization

- *Visualization of individual attributes with histograms*

Code:

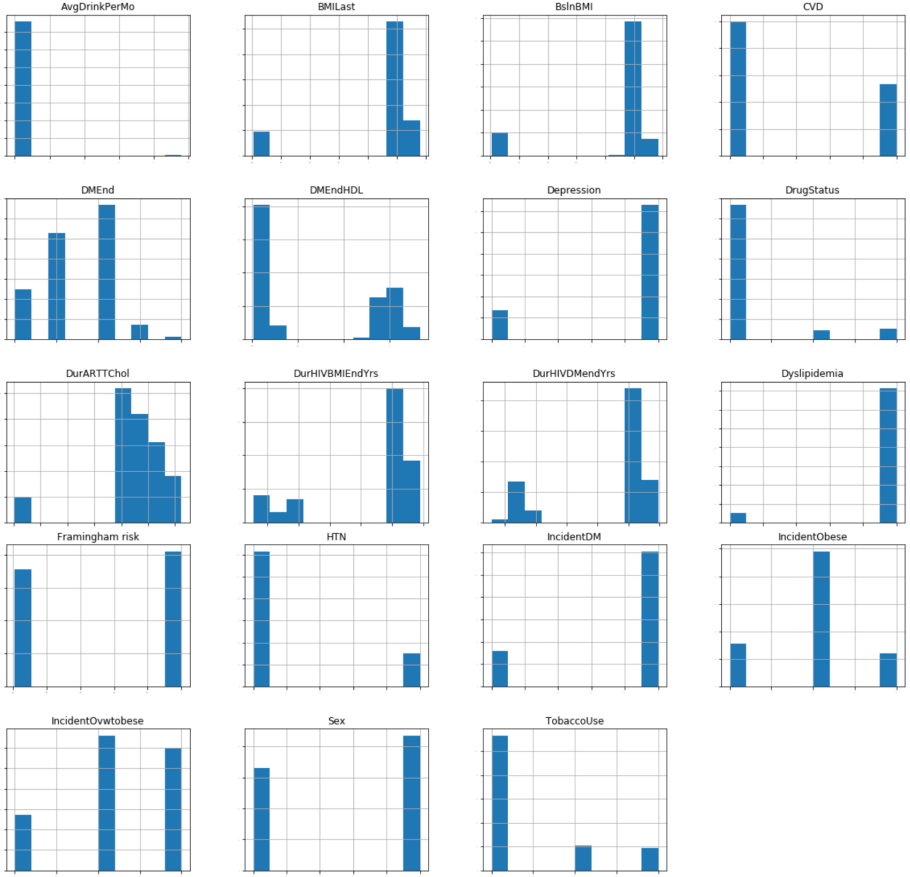
```
# histograms
dataset.hist(sharex=False, sharey=False, xlabelsize=1, ylabelsize=1)
pyplot.rcParams["figure.figsize"] = (20,40)

pyplot.show()
```

2 – Example - Classification

Data Understanding

Phase I - Data Understanding – with Visualization

Output:



2 – Example - Classification

Data Understanding

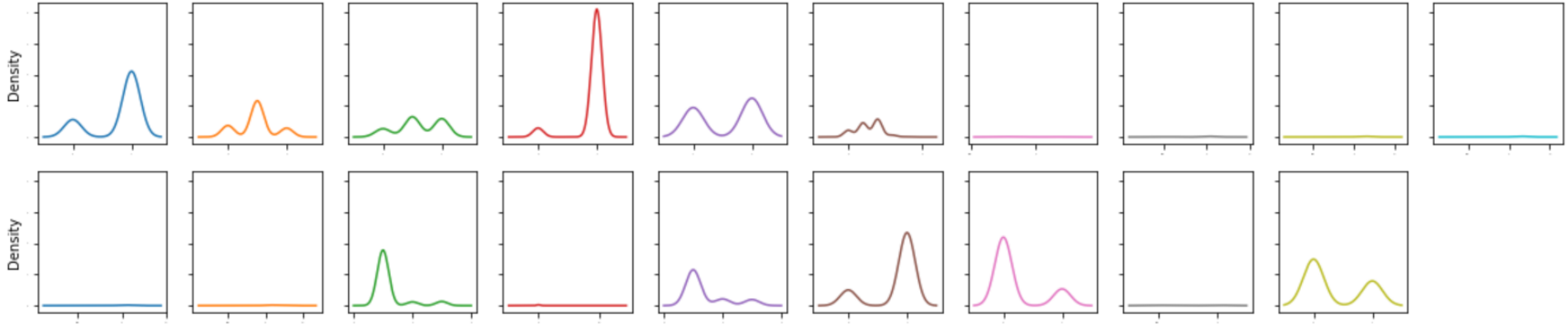
Phase I - Data Understanding – with Visualization

- *Visualization of individual attributes with density plots*

Code:

```
# density
dataset.plot(kind='density', subplots=True, layout=(9,10), sharex=False,sharey=True, legend=False, fontsize=1)
pyplot.rcParams["figure.figsize"] = (20,20)
pyplot.show()
```

Output:



2 – Example - Classification

Data Understanding

Phase I - Data Understanding – with Visualization

- *Visualize correlation between attributes*

Code:

```
# correlation matrix
column=dataset.columns
fig = pyplot.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(dataset.corr('pearson'), vmin=-1, vmax=1, interpolation='none')
ax.set_xticks(np.arange(len(column)))
ax.set_yticks(np.arange(len(column)))
ax.set_xticklabels(column)
ax.set_yticklabels(column)
fig.colorbar(cax)
pyplot.rcParams["figure.figsize"] = (10,20)
pyplot.show()
```

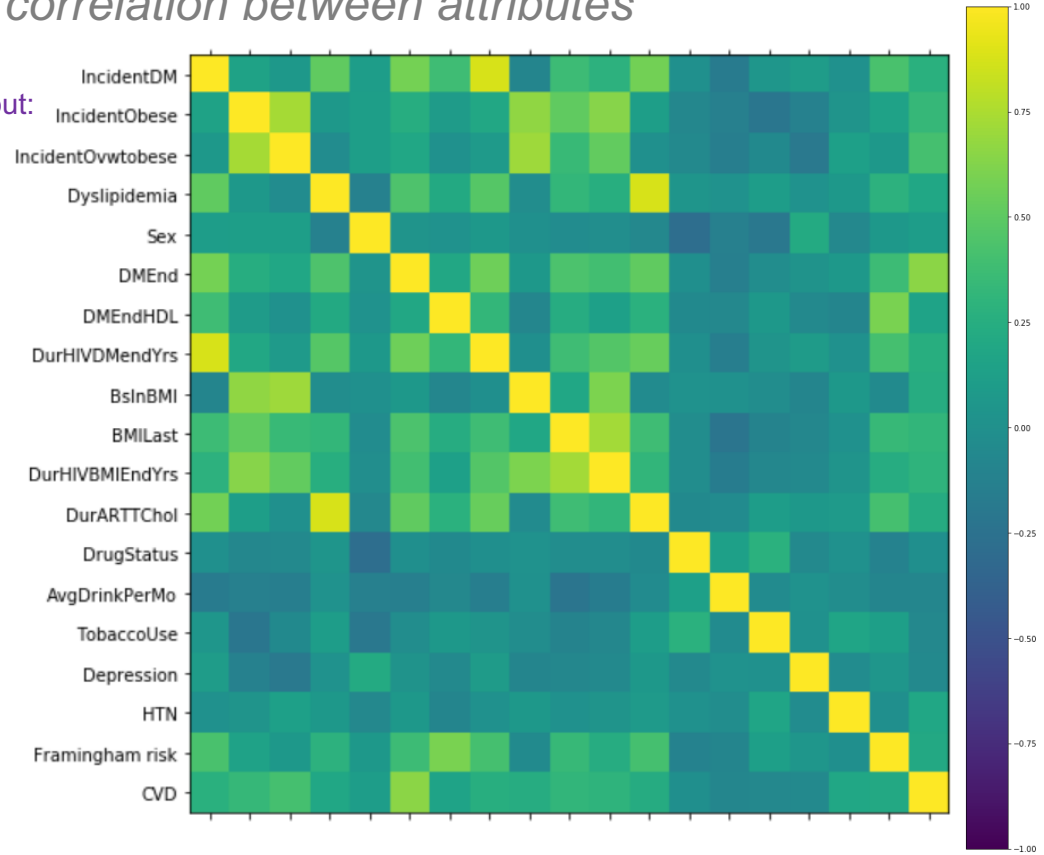
2 – Example - Classification

Data Understanding

Phase I - Data Understanding – with Visualization

- *Visualize correlation between attributes*

Output:



2 – Example - Classification

Data Preprocessing

Phase II - Data Preprocessing – Data Transformation

- *Standardization*

Code:

```
from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions
filename = "CVD_in_ART.csv"
dataframe = read_csv(filename,  index_col='Patient ID')
array = dataframe.values
X = array[:,0:11]
Y = array[:,11]
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
# summarize transformed data
set_printoptions(precision=2)
print(rescaledX[0:3,:])
```

Output:

```
[[ 5.14e-01  1.75e+00  1.09e+00  2.64e-01  8.71e-01  7.37e-01  4.09e-01
   5.83e-01  4.14e-01 -3.55e-01 -7.60e-02]
 [-1.94e+00  7.65e-02  1.09e+00  2.64e-01  8.71e-01  7.37e-01  3.78e-01
   4.09e-01  3.83e-01 -3.55e-01  1.35e-03]
 [ 5.14e-01  1.75e+00  1.09e+00  2.64e-01  8.71e-01  7.37e-01  4.09e-01
   5.34e-01  4.14e-01 -3.55e-01 -1.53e-01]]
```

2 – Example - Classification

Model Planning

Phase III – Model Planning – Split-out dataset

- ***Validation hold-out set*** *is a sample of the data that is hold back from the analysis and modeling (in the example, we hold back 20% for validation)*

Code:

```
# Split-out validation dataset
array = dataset.values
X = array[:,0:11]
Y = array[:,11]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size, random_state=seed)
```

2 – Example - Classification

Model Planning

Phase III – Model Planning – Performance evaluation

- *k-fold Cross-Validation*

Code:

```
# Test options and evaluation metric
num_folds = 10
seed = 7
```

2 – Example - Classification

Model Planning

Phase III – Model Planning – Performance metrics

- *Accuracy*

Code:

```
scoring = 'accuracy'
```

2 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – using raw dataset*

Code:

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X_train, Y_train, cv=10, scoring="accuracy")
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

2 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – using raw dataset*

Output:

```
LR: 0.802727 (0.101073)
LDA: 0.802727 (0.119230)
KNN: 0.771818 (0.156799)
CART: 0.881818 (0.087670)
NB: 0.634545 (0.143152)
SVM: 0.792727 (0.071904)
```

2 – Example - Classification

Model Building

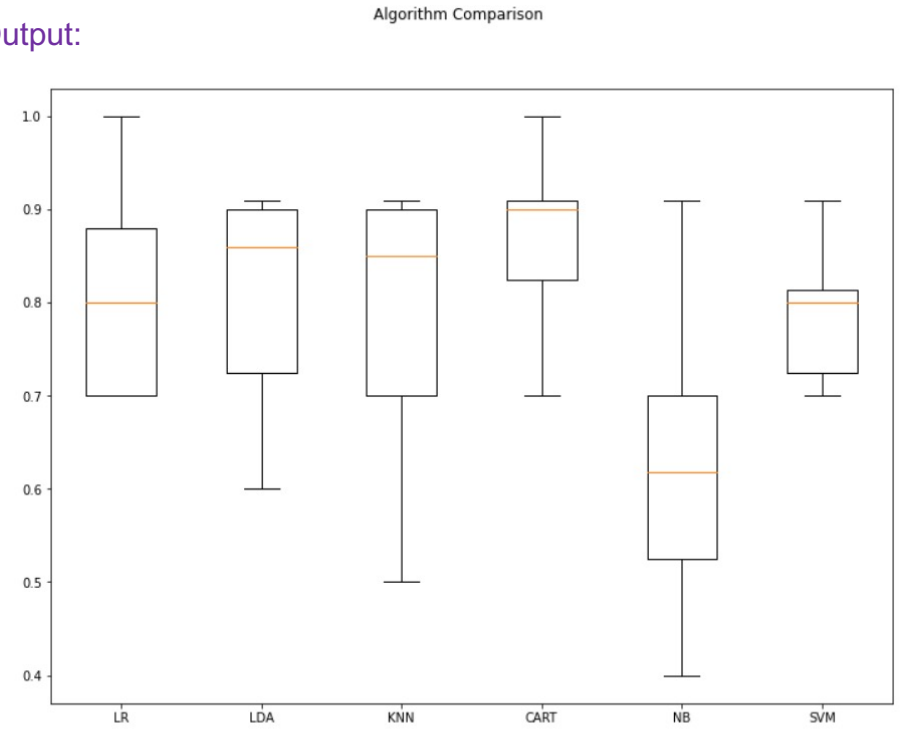
Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – visualization using raw dataset*

Code:

```
# Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.rcParams["figure.figsize"] = (12,9)
pyplot.show()
```

Output:



2 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – using scaled/standardized dataset*

Code:

```
# Standardize the dataset
scaler = StandardScaler().fit(X_train)
X_scaled = scaler.transform(X_train)
models = []
models.append(('LR', LogisticRegression(solver='liblinear')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X_scaled, Y_train, cv=10, scoring="accuracy")
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

2 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – using scaled/standardized dataset*

Output:

```
LR: 0.810909 (0.151892)
LDA: 0.802727 (0.119230)
KNN: 0.705455 (0.125586)
CART: 0.910909 (0.094283)
NB: 0.655455 (0.125603)
SVM: 0.607273 (0.014545)
```

2 – Example - Classification

Model Building

Phase IV – Model Building – Algorithm Tunning

- *Grid Search Parameter Tuning – on CART(best models)*

Code:

```
# Tune scaled CART
scaler = StandardScaler().fit(X_validation)
rescaledX = scaler.transform(X_validation)
param_grid =   {
'max_depth': [2, 4,8, 16],
'min_samples_leaf': [0.01, 0.02, 0.03],
'max_features': [0.2, 0.4,0.6, 0.8]
}

model =  DecisionTreeClassifier()

kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=True)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=kfold)
grid_result = grid.fit(rescaledX, Y_validation)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Output:

```
Best: 0.980000 using {'max_depth': 4, 'max_features': 0.6, 'min_samples_leaf': 0.02}
0.720000 (0.271293) with: {'max_depth': 2, 'max_features': 0.2, 'min_samples_leaf': 0.01}
0.780000 (0.244131) with: {'max_depth': 2, 'max_features': 0.2, 'min_samples_leaf': 0.02}
0.763333 (0.175404) with: {'max_depth': 2, 'max_features': 0.2, 'min_samples_leaf': 0.03}
0.870000 (0.200250) with: {'max_depth': 2, 'max_features': 0.4, 'min_samples_leaf': 0.01}
0.770000 (0.268514) with: {'max_depth': 2, 'max_features': 0.4, 'min_samples_leaf': 0.02}
0.780000 (0.188680) with: {'max_depth': 2, 'max_features': 0.4, 'min_samples_leaf': 0.03}
0.820000 (0.140000) with: {'max_depth': 2, 'max_features': 0.6, 'min_samples_leaf': 0.01}
0.900000 (0.184391) with: {'max_depth': 2, 'max_features': 0.6, 'min_samples_leaf': 0.02}
0.880000 (0.160000) with: {'max_depth': 2, 'max_features': 0.6, 'min_samples_leaf': 0.03}
0.900000 (0.161245) with: {'max_depth': 2, 'max_features': 0.8, 'min_samples_leaf': 0.01}
0.880000 (0.203961) with: {'max_depth': 2, 'max_features': 0.8, 'min_samples_leaf': 0.02}
0.900000 (0.134164) with: {'max_depth': 2, 'max_features': 0.8, 'min_samples_leaf': 0.03}
0.843333 (0.149108) with: {'max_depth': 4, 'max_features': 0.2, 'min_samples_leaf': 0.01}
0.780000 (0.188680) with: {'max_depth': 4, 'max_features': 0.2, 'min_samples_leaf': 0.02}
0.903333 (0.132035) with: {'max_depth': 4, 'max_features': 0.2, 'min_samples_leaf': 0.03}
0.886667 (0.149220) with: {'max_depth': 4, 'max_features': 0.4, 'min_samples_leaf': 0.01}
0.880000 (0.160000) with: {'max_depth': 4, 'max_features': 0.4, 'min_samples_leaf': 0.02}
0.820000 (0.166132) with: {'max_depth': 4, 'max_features': 0.4, 'min_samples_leaf': 0.03}
0.903333 (0.132035) with: {'max_depth': 4, 'max_features': 0.6, 'min_samples_leaf': 0.01}
0.980000 (0.060000) with: {'max_depth': 4, 'max_features': 0.6, 'min_samples_leaf': 0.02}
0.866667 (0.146059) with: {'max_depth': 4, 'max_features': 0.6, 'min_samples_leaf': 0.03}
0.940000 (0.128062) with: {'max_depth': 4, 'max_features': 0.8, 'min_samples_leaf': 0.01}
0.943333 (0.086987) with: {'max_depth': 4, 'max_features': 0.8, 'min_samples_leaf': 0.02}
0.860000 (0.200998) with: {'max_depth': 4, 'max_features': 0.8, 'min_samples_leaf': 0.03}
0.763333 (0.216256) with: {'max_depth': 8, 'max_features': 0.2, 'min_samples_leaf': 0.01}
0.870000 (0.155242) with: {'max_depth': 8, 'max_features': 0.2, 'min_samples_leaf': 0.02}
0.823333 (0.188591) with: {'max_depth': 8, 'max_features': 0.2, 'min_samples_leaf': 0.03}
0.880000 (0.132665) with: {'max_depth': 8, 'max_features': 0.4, 'min_samples_leaf': 0.01}
0.920000 (0.132665) with: {'max_depth': 8, 'max_features': 0.4, 'min_samples_leaf': 0.02}
0.840000 (0.195959) with: {'max_depth': 8, 'max_features': 0.4, 'min_samples_leaf': 0.03}
0.920000 (0.132665) with: {'max_depth': 8, 'max_features': 0.6, 'min_samples_leaf': 0.01}
0.920000 (0.183303) with: {'max_depth': 8, 'max_features': 0.6, 'min_samples_leaf': 0.02}
0.900000 (0.134164) with: {'max_depth': 8, 'max_features': 0.6, 'min_samples_leaf': 0.03}
0.900000 (0.184391) with: {'max_depth': 8, 'max_features': 0.8, 'min_samples_leaf': 0.01}
0.960000 (0.080000) with: {'max_depth': 8, 'max_features': 0.8, 'min_samples_leaf': 0.02}
0.900000 (0.134164) with: {'max_depth': 8, 'max_features': 0.8, 'min_samples_leaf': 0.03}
0.766667 (0.169312) with: {'max_depth': 16, 'max_features': 0.2, 'min_samples_leaf': 0.01}
0.823333 (0.139881) with: {'max_depth': 16, 'max_features': 0.2, 'min_samples_leaf': 0.02}
0.820000 (0.166132) with: {'max_depth': 16, 'max_features': 0.2, 'min_samples_leaf': 0.03}
0.823333 (0.208726) with: {'max_depth': 16, 'max_features': 0.4, 'min_samples_leaf': 0.01}
0.860000 (0.200998) with: {'max_depth': 16, 'max_features': 0.4, 'min_samples_leaf': 0.02}
0.763333 (0.175404) with: {'max_depth': 16, 'max_features': 0.4, 'min_samples_leaf': 0.03}
0.940000 (0.091652) with: {'max_depth': 16, 'max_features': 0.6, 'min_samples_leaf': 0.01}
0.960000 (0.080000) with: {'max_depth': 16, 'max_features': 0.6, 'min_samples_leaf': 0.02}
0.860000 (0.128062) with: {'max_depth': 16, 'max_features': 0.6, 'min_samples_leaf': 0.03}
0.940000 (0.128062) with: {'max_depth': 16, 'max_features': 0.8, 'min_samples_leaf': 0.01}
0.940000 (0.128062) with: {'max_depth': 16, 'max_features': 0.8, 'min_samples_leaf': 0.02}
0.920000 (0.132665) with: {'max_depth': 16, 'max_features': 0.8, 'min_samples_leaf': 0.03}
```

Best result: ***accuracy = 0.98 -*** *parameters max_depth = 4, min_simples_leaf = 0.02, max_features = 0.6*

2 – Example - Classification

Model Building

Phase IV – Model Building – Algorithm Tunning

- *Grid Search Parameter Tuning – on LR (best models)*

Code:

```
# Tune scaled LR
scaler = StandardScaler().fit(X_validation)
rescaledX = scaler.transform(X_validation)
param_grid =  {
     'penalty' : [ 'l2'],
    'C' : [1.00000000e-04 , 2.63665090e-04, 6.95192796e-04 ,1.83298071e-03,
 4.83293024e-03, 1.27427499e-02,3.35981829e-02, 8.85866790e-02,
 2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
 1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
 5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04] }

model = LogisticRegression(max_iter=3000)

kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=True)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=kfold)
grid_result = grid.fit(rescaledX, Y_validation)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Output:

```
Best: 0.940000 using {'C': 545.559478, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.0001, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.00026366509, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.000695192796, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.00183298071, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.00483293024, 'penalty': 'l2'}
0.720000 (0.256125) with: {'C': 0.0127427499, 'penalty': 'l2'}
0.760000 (0.280000) with: {'C': 0.0335981829, 'penalty': 'l2'}
0.840000 (0.249800) with: {'C': 0.088586679, 'penalty': 'l2'}
0.900000 (0.184391) with: {'C': 0.233572147, 'penalty': 'l2'}
0.900000 (0.134164) with: {'C': 0.615848211, 'penalty': 'l2'}
0.900000 (0.134164) with: {'C': 1.62377674, 'penalty': 'l2'}
0.900000 (0.134164) with: {'C': 4.2813324, 'penalty': 'l2'}
0.900000 (0.134164) with: {'C': 11.2883789, 'penalty': 'l2'}
0.920000 (0.132665) with: {'C': 29.7635144, 'penalty': 'l2'}
0.920000 (0.132665) with: {'C': 78.475997, 'penalty': 'l2'}
0.920000 (0.132665) with: {'C': 206.913808, 'penalty': 'l2'}
0.940000 (0.128062) with: {'C': 545.559478, 'penalty': 'l2'}
0.940000 (0.128062) with: {'C': 1438.44989, 'penalty': 'l2'}
0.940000 (0.128062) with: {'C': 3792.69019, 'penalty': 'l2'}
0.940000 (0.128062) with: {'C': 10000.0, 'penalty': 'l2'}
```

Best result: ***accuracy = 0.94 -*** *parameters C=545.56, penalty = 12*

2 – Example - Classification

Results

Phase V – Results – Finalize model and predict on validation dataset

- *Training the **selected algorithm (CART)** with **parameters** (i.e. max_depth = 4, min_samples_leaf = 0.02, max_features = 0.6) now in the **entire dataset (without cross-validation)** and **validate** (compute the metrics) for the hold-out validation dataset to communicate this results.*

2 – Example - Classification

Results

Phase V – Results – Finalize model and predict on validation dataset

Code:

```
# Finalize Model

# prepare the model
scaler = StandardScaler().fit(X_validation)
rescaledX = scaler.transform(X_validation)
model = DecisionTreeClassifier(max_depth=2, max_features= 0.6, min_samples_leaf =0.02)
model.fit(rescaledX, Y_validation)
# estimate accuracy on validation dataset
rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

Output:

```
Accuracy 0.9607843137254902
[[37  0]
 [ 2 12]]
              precision    recall  f1-score   support

         0.0       0.95      1.00      0.97        37
         1.0       1.00      0.86      0.92        14

    accuracy                           0.96        51
   macro avg       0.97      0.93      0.95        51
weighted avg       0.96      0.96      0.96        51
```

Results

2 – Example - Classification

Results

Phase V – Results – Save model

- *Save model with pickle*

Code:

```
from pickle import dump
from pickle import load
# save the model to disk
model_file = 'model_file.sav'
dump(model, open(model_file, 'wb'))
```

- *Save model with joblib*

Code:

```
from joblib import dump
from joblib import load
# save the model to disk
model_file = 'model_file.sav'
dump(model, open(model_file, 'wb'))
```

Unit 10: Final Project - Demonstration and Instructions- Python

1 – Project Template

2 – Example - Classification

3 – Project Orientation

3 – Final Project- Classification

Introduction - Define the problem

- *Hepatitis C is a liver infection caused by the hepatitis C virus (HCV).*
- *For more than half of people who become infected with the hepatitis C virus, it becomes a long-term, chronic infection which can result in serious, even life-threatening health problems like cirrhosis and liver cancer.*
- *Hepatitis C is spread through contact with blood from an infected person. The best way to prevent hepatitis C is by avoiding behaviors that can spread the disease.*
- *This project **aims** to create a model to predict if patients will be infected with Hepatitis C based on their demographics and regular laboratory test results.*
- *The dataset contains data from Puerto Rican patients who have been suspected of having Hepatitis C based on their regular laboratory results and have taken an Ab o RNA test to confirm it. The **data** is in the file "Hepatitis C.csv".*

3 - Project Template

To complete the project

```
# Load the library that is used to manipulate dataframes

# Load dataset
```

```
# Print the shape of the dataset

# Print  the summary description of the dataset

# Print the first 10 lines of the dataset
```

```
'''
Using the functions groupby and size,
determine how many patients have been
diagnosed with Hepatitis C and how many are control cases.
'''
```

Data Visualization

```
# Import the pyplot fuction from the matplotlib library

# Plot the histogram of the data

# Display the plot
```

```
# Plot the correlation matrix of the data
```

Data Preparation

```
# Using the values function, convert the dataset into an array and store it into a variable

# Separate the target variable from the rest of the array using positional  indexing

# Import the train_test_split from  sklearn.model_selection

# Using the function train_test_split split-out validation and training dataset
```

- *How to proceed in Jupyter:*
  - *Load the notebook in Jupyter named “Project_Template” using Python 3*
  - *Start to fill it in using the project example as a model*

3 – Example - Classification

Introduction - Load Dataset

```
# Load the library that is used to manipulate dataframes


# Load dataset and store it in the variable dataset
```

3 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

```
#Print the shape of the dataset


#Print  the summary description of the dataset


#Print the first 10 lines of the dataset
```

3 – Example - Classification

Data Understanding

Phase I - Data Understanding - with statistics

- *Breakdown of class values*

```
''''
Using the functions groupby and size,
determine how many patients have been
diagnosed with Hepatitis C and how many are control cases.
''''
```

3 – Example - Classification

Data Understanding

Phase I - Data Understanding – with Visualization

```
# Import the pyplot fuction from the matplotlib library


# Plot the histogram of the data using the hist function


# Display the plot using the show function
```

3 – Example - Classification

Data Preprocessing

Phase II - Data Preprocessing – Data Transformation

```
# Using the values function, convert the dataset into an array and store it into a variable

# Separate the target variable from the rest of the array using positional  indexing

#Import the train_test_split from  sklearn.model_selection

# Using the function train_test_split split-out validation and training dataset
```

3 – Example - Classification

Model Planning

Phase III – Model Planning – Split-out dataset

```
# Using the values function, convert the dataset into an array and store it into a variable

# Separate the target variable from the rest of the array using positional  indexing

#Import the train_test_split from  sklearn.model_selection

# Using the function train_test_split split-out validation and training dataset
```

3 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms*

```
# Import the  LogisticRegression function from sklearn.linear_model

# Import the DecisionTreeClassifier function from sklearn.tree

# Import the KNeighborsClassifier function from sklearn.neighbors

# Import the LinearDiscriminantAnalysis function from sklearn.discriminant_analysis

# Import the GaussianNB function from sklearn.naive_bayes

# Import the SVC function from sklearn.svm
```

3 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms*

```
# Create an empty list called model


# Using the append function, add an alias for the model and the  function to the list,
# using "liblinear" as the solver.
# This one has been done for you.
# models.append(('LR', Logistic Regression(solver='liblinear')))


# Using the append function, add an alias for the Linear Discriminant Analysis model and the
# LinearDiscriminantAnalysis function to the list.


# Using the append function, add an alias for the K Neighbors Classifier model and the
# KNeighborsClassifier function to the list.


# Using the append function, add an alias for the Decision Tree Classifier model and the
# DecisionTreeClassifier function to the list.


# Using the append function, add an alias for the Gaussian Naive Bayes model and the
# GaussianNB function to the list.


# Using the append function, add an alias for the Support Vector Classifier model and the
# SVC function to the list,using 'auto' as the value for the gamma parameter.


# Create an empty list called results


# Create an empty list called names
```

3 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – This part has been done for you*

```
# Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.rcParams["figure.figsize"] = (12,9)
pyplot.show()
```

3 – Example - Classification

Model Building

Phase IV – Model Building – Training and Comparing

- *Comparison of Classification Algorithms – This part has been done for you*

```
# Import StandardScaler from sklearn.preprocessing


# Create an instance of  StandardScaler and store it into a variable called scaler


# Fit the training data to the scaler by employing the fit function


# Transform the training data to the scaler by employing the transform function and store it into a variable called
# S_scaled
```

3 – Example - Classification

Model Building

Phase IV – Model Building – Algorithm Tunning

```
# If the best performing group of models were the scaled models, scale the validation data
# If not, leave this space blank


# Create an instance of the Best performing model and store it into the variabe model


# Create a grid of parameters to test for the Best performing model
# In the jupyter template for Project example there are parameters available for each of the models


# Import KFold from sklearn.model_selection


# Create an instance of Kfold with the following parameters: n_splits=num_folds, random_state=seed, shuffle=True
# Store it into a variable called kfold


# Import GridSearchCV from sklearn.model_selection

# Create an instance of GridSearchCV with the following parameters: estimator=model, param_grid=param_grid,
# scoring=scoring, cv=kfold. Store it into a variable called grid


# Fit the validation data to the grid by employing the fit function, store it into a variable called grid_result


# The following code will print the results of the grid search
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

3 – Example - Classification

Results

Phase V – Results – Finalize model and predict on validation dataset

```
# If the best performing models after modeling was the scaled, scale the validation data
#If not, leave this space blank


# Create an instance of the best performing model and store it into the variabe model


# Fit the validation data to the model by employing the fit function


# Estimate accuracy on validation dataset by employing the fit function on model
# store it into a variable called predictions


# Import accuracy_score from sklearn.metrics


# Import confusion_matrix from sklearn.metrics


# Import classification_report from sklearn.metrics


#Inside a print statement, call the fuction accuracy_score, with the following parameters:
#Y_validation, predictions


#Inside a print statement, call the fuction confusion_matrix, with the following parameters:
#Y_validation, predictions


#Inside a print statement, call the fuction classification_report, with the following parameters:
#Y_validation, predictions
```

The End!!

Unit 10: Final Project - Demonstration and Instructions- Python