Project Assisting Reports

- Report 1
  Feature Statistical Testing Report

  page (02 – 05)

- Report 2
  Models Assessment Report

  page (06 - 07)

- Report 3
  Testing and Comparing the effects of scaled and unscaled data on the Logistic Regression Model

  page (08 - 13)

- Report 4
  Models Built Report

  page (14 - 16)

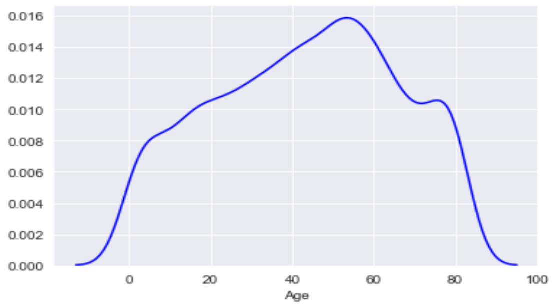
Report 1

Dataset Feature Statistical Testing Report

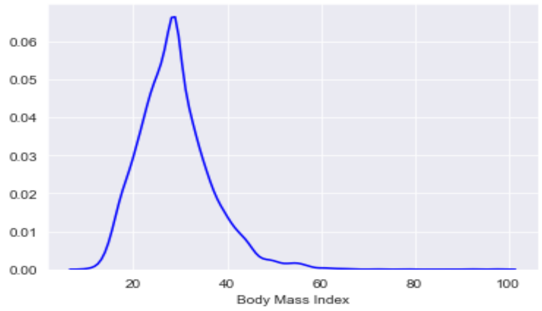
NUMERIC INDEPENDENT FEATURE STATISTICAL TESTING & ANALYSIS

Shapiro Wilk Test

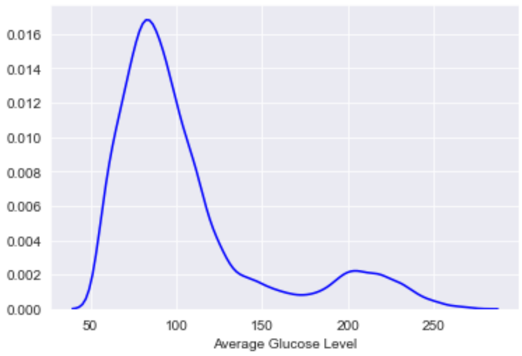
```
Age
stat=0.968, p=0.000
Age variable probable not to follow a Gaussian Distribution
```




```
Body Mass Index
stat=0.952, p=0.000
Body Mass Index variable probable not to follow a Gaussian Distribution
```




```
Average Glucose Level
stat=0.806, p=0.000
Average Glucose Level variable probable not to follow a Gaussian Distri
bution
```



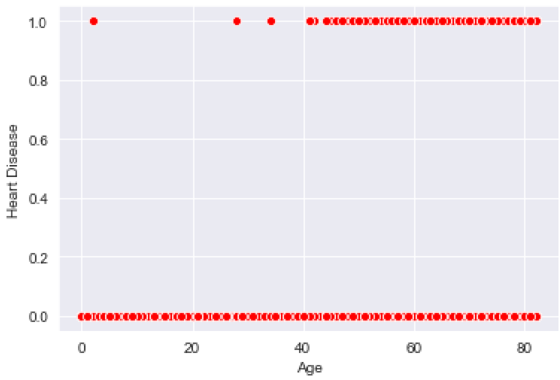
Pearsons Correlation Coefficient Test

	Age	Average Glucose Level	Body Mass Index
Age	1.000000	0.238060	0.326284
Average Glucose Level	0.238060	1.000000	0.168767
Body Mass Index	0.326284		1.000000

NUMERIC **DEPENDENT** FEATURE ANALYSIS

Scatterplot

Heart Diseases (Dependent Variable) / Age



CATEGORICAL INDEPENDENT FEATURE STATISTICAL TESTING & ANALYSIS

Chi Square Statistical Testing

```
Gender

stat=6.559, p=0.038
Gender and Marriage Status Probable Dependent

stat=2.410, p=0.300
Gender and Hypertension Probable Independent

stat=43.075, p=0.000
Gender and Work Type Probable Dependent

stat=1.223, p=0.542
Gender and Residence Type Probable Independent

stat=57.338, p=0.000
Gender and Smoking Status Probable Dependent

stat=0.473, p=0.790
Gender and Stroke Probable Independent
```

```
Marriage Status

stat=6.559, p=0.038
Gender and Marriage Status Probable Dependent

stat=136.683, p=0.000
Marriage Status and Hypertension Probable Dependent

stat=1644.109, p=0.000
Marriage Status and Work Type Probable Dependent

stat=0.175, p=0.676
Marriage Status and Residence Type Probable Independent

stat=599.046, p=0.000
Marriage Status and Smoking Status Probable Dependent

stat=58.924, p=0.000
Marriage Status and Stroke Probable Dependent
```

```
Hypertension

stat=136.683, p=0.000
Marriage Status and Hypertension Probable Dependent

stat=2.410, p=0.300
Gender and Hypertension Probable Independent

stat=135.200, p=0.000
```

```
Hypertension & Work Type Probable Dependent

stat=0.269, p=0.604
Hypertension & Residence Type Probable Independent

stat=103.874, p=0.000
Hypertension & Smoking Status Probable Dependent

stat=81.605, p=0.000
Hypertension & Stroke Probable Dependent
```

```
Work Type

stat=135.200, p=0.000
Hypertension & Work Type Probable Dependent

stat=43.075, p=0.000
Gender and Work Type Probable Dependent

stat=4.653, p=0.325
Work Type & Residence Type Probable Independent

stat=1644.109, p=0.000
Marriage Status and Work Type Probable Dependent

stat=1389.107, p=0.000
Work Type & Smoking Status Probable Dependent

stat=49.164, p=0.000
Work Type & Stroke Probable Dependent
```

Report 2

Models Assessment Report

Six Model Accuracy Scores

Model One	
Accuracy	0.9491392801251957
Model Two	
Accuracy	0.7613458528951487
Model Three	
Accuracy	0.9483568075117371
Model Four	
Accuracy	0.9491392801251957
Model Five	
Accuracy	0.6697965571205008
Model Six	
Accuracy	0.7566510172143975

Six Model Classification Reports

Model One

	precision	recall	f1-score	support
class 0	0.95	1.00	0.97	1213
class 1	***0.50***	***0.05***	***0.08***	***65***

Model Two

	precision	recall	f1-score	support
class 0	0.99	0.76	0.86	1213
class 1	**0.15**	**0.78**	**0.25**	**65**

Model Three

	precision	recall	f1-score	support
class 0	0.95	1.00	0.97	1213
class 1	**0.43**	**0.05**	**0.08**	**65**

Model Four

	precision	recall	f1-score	support
class 0	0.95	1.00	0.97	1213
class 1	**0.50**	**0.03**	**0.06**	**65**

Model Five

	precision	recall	f1-score	support

	precision	recall	f1-score	support
class 0	0.98	0.67	0.79	1213
class 1	**0.10**	**0.71**	**0.18**	**65**

Model Six

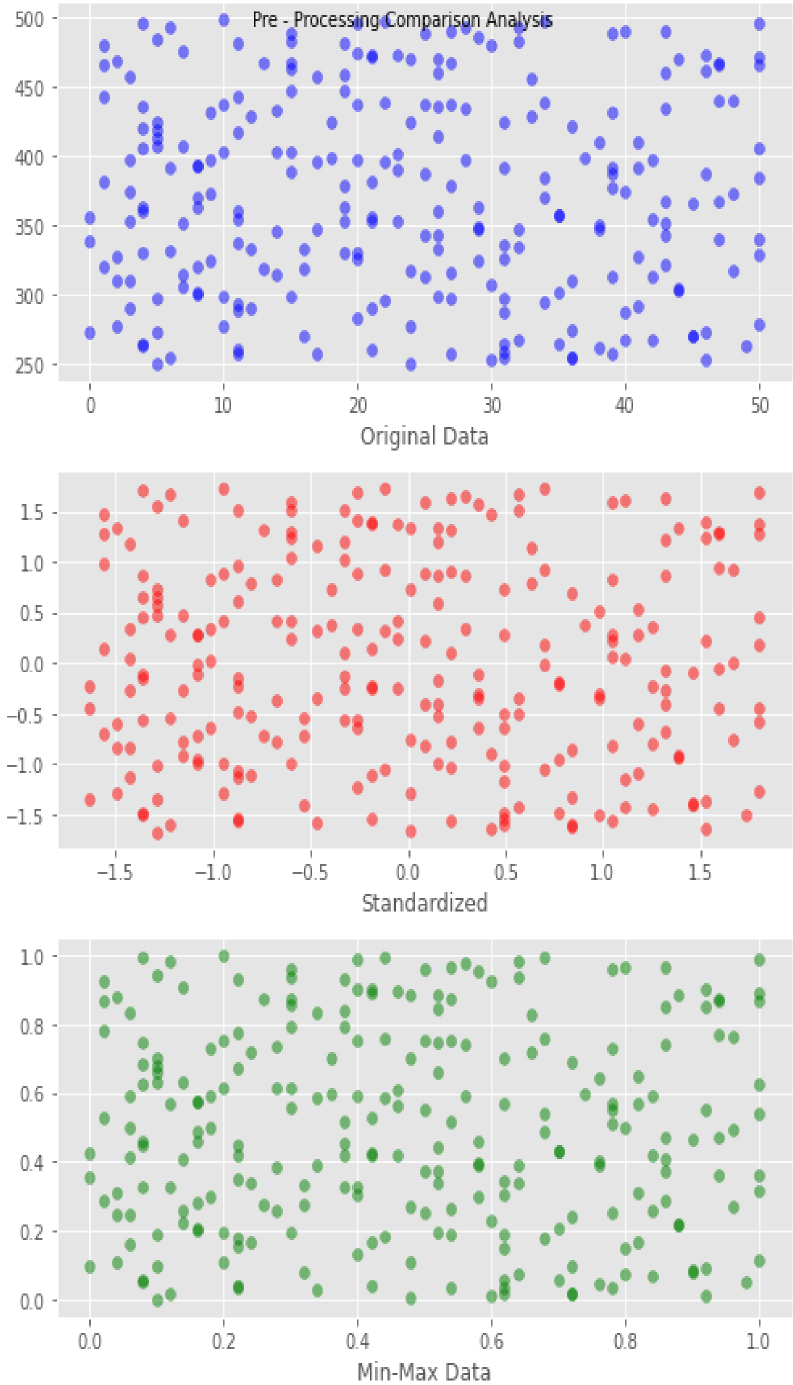
	precision	recall	f1-score	support
class 0	0.99	0.75	0.85	1213
class 1	0.15	0.80	0.25	65

Report 3

Testing and comparing the effects of scaled and unscaled data on the Logistic Regression Algorithm

Test 1

Dataset Shape (250, 7)



Train Test Split 0.5

Original Data

```
[[76  3]

 [46  0]]
              precision    recall  f1-score   support

     class 0       0.62      0.96      0.76        79
     class 1       0.00      0.00      0.00        46

    accuracy                           0.61       125
   macro avg       0.31      0.48      0.38       125
weighted avg       0.39      0.61      0.48       125

Accuracy: 0.608
```

Standardized Data

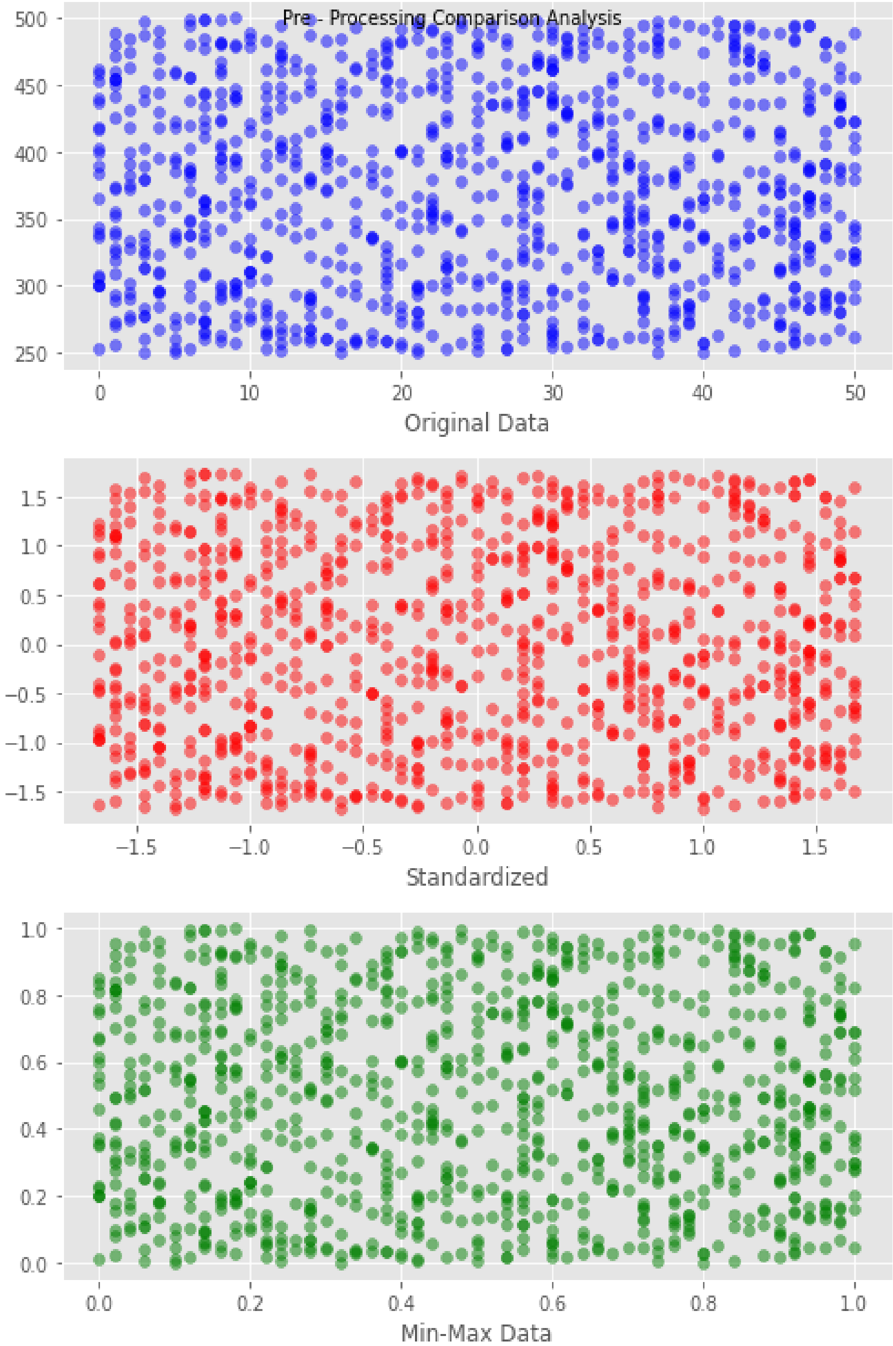
```
[[78  1]
 [46  0]]
              precision    recall  f1-score   support

     class 0       0.63      0.99      0.77        79
     class 1       0.00      0.00      0.00        46

    accuracy                           0.62       125
   macro avg       0.31      0.49      0.38       125
weighted avg       0.40      0.62      0.49       125

Accuracy: 0.624
```

Min-Max Scaled Data

```
[[79  0]
 [46  0]]
              precision    recall  f1-score   support

     class 0       0.63      1.00      0.77        79
     class 1       0.00      0.00      0.00        46

    accuracy                           0.63       125
   macro avg       0.32      0.50      0.39       125
weighted avg       0.40      0.63      0.49       125

Accuracy: 0.632
```

Test 2

Dataset Shape (1000, 7)



Train Test Split 0.6

Original Data

```
[[274   6]
 [118   2]]
              precision    recall  f1-score   support

     class 0       0.70      0.98      0.82       280
     class 1       0.25      0.02      0.03       120

    accuracy                           0.69       400
   macro avg       0.47      0.50      0.42       400
weighted avg       0.56      0.69      0.58       400

Accuracy: 0.69
```

Standardized Data

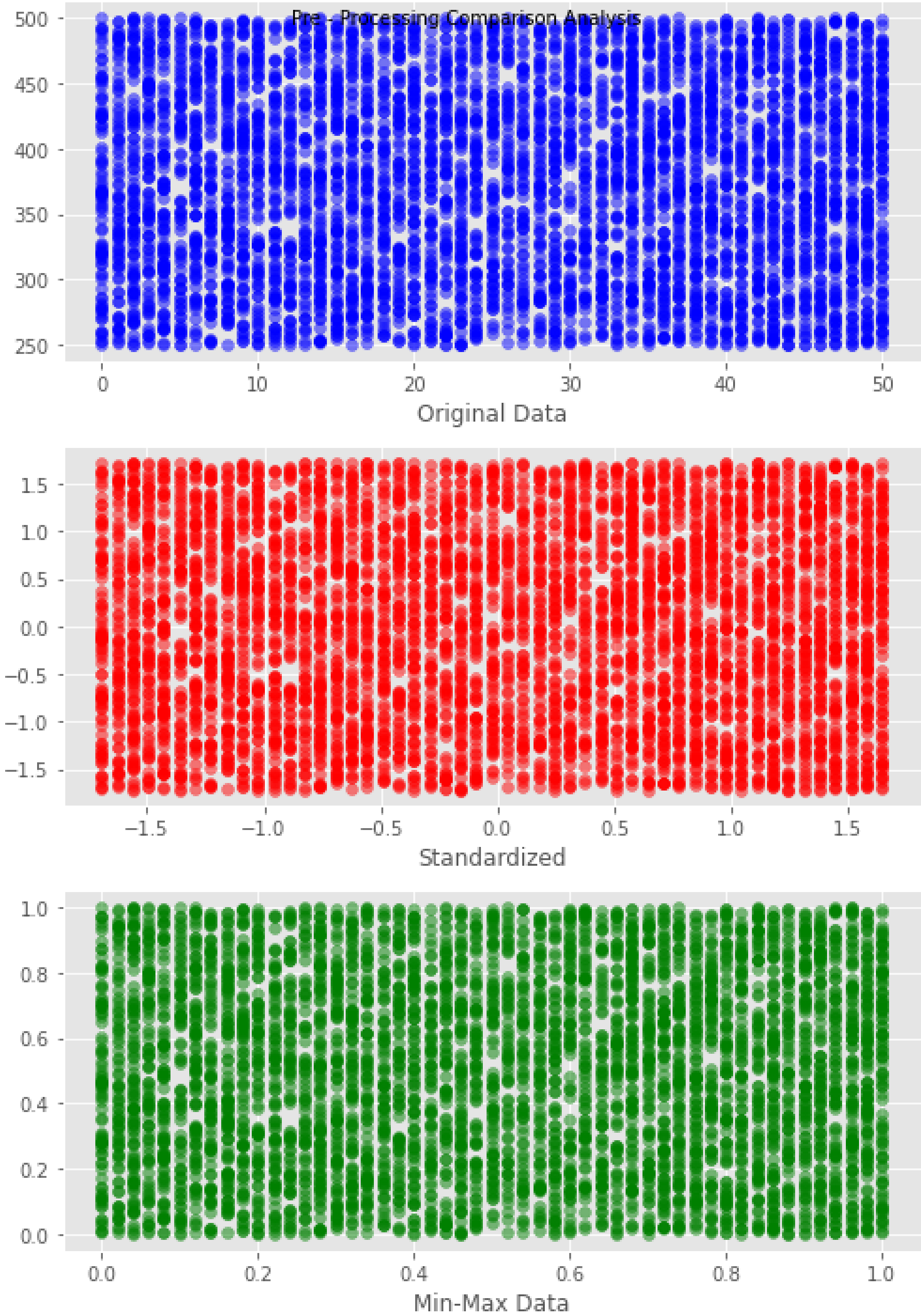
```
[[278   2]
 [120   0]]
              precision    recall  f1-score   support

     class 0       0.70      0.99      0.82       280
     class 1       0.00      0.00      0.00       120

    accuracy                           0.69       400
   macro avg       0.35      0.50      0.41       400
weighted avg       0.49      0.69      0.57       400

Accuracy: 0.695
```

Min Max Scaled Data

```
[[278   2]
 [120   0]]
              precision    recall  f1-score   support

     class 0       0.70      0.99      0.82       280
     class 1       0.00      0.00      0.00       120

    accuracy                           0.69       400
   macro avg       0.35      0.50      0.41       400
weighted avg       0.49      0.69      0.57       400

Accuracy: 0.695
```

Test 3

Dataset Shape (5000, 7)



Train Test Split 0.7

Original Data

```
[[1069    0]
 [ 431    0]]
              precision    recall  f1-score   support

     class 0       0.71      1.00      0.83      1069
     class 1       0.00      0.00      0.00       431

    accuracy                           0.71      1500
   macro avg       0.36      0.50      0.42      1500
weighted avg       0.51      0.71      0.59      1500

Accuracy: 0.7126666666666667
```

Standardized Data

```
[[1069    0]
 [ 431    0]]
              precision    recall  f1-score   support

     class 0       0.71      1.00      0.83      1069
     class 1       0.00      0.00      0.00       431

    accuracy                           0.71      1500
   macro avg       0.36      0.50      0.42      1500
weighted avg       0.51      0.71      0.59      1500

Accuracy: 0.7126666666666667
```

Min Max Scaled Data

```
[[1069    0]
 [ 431    0]]
              precision    recall  f1-score   support

     class 0       0.71      1.00      0.83      1069
     class 1       0.00      0.00      0.00       431

    accuracy                           0.71      1500
   macro avg       0.36      0.50      0.42      1500
weighted avg       0.51      0.71      0.59      1500

Accuracy: 0.7126666666666667
```

Model One

Building Model

```
# Build a Logisitc Regression Model (using the default parameters)
Model_One = LogisticRegression()

# fit the model with data (Training Model)
Model_One.fit(X_train,y_train)

# Prediction using model (test Model)
Model_Predictions = Model_One.predict(X_test)
```

Model Two

Building Model

```
# Build a Logisitc Regression Model (Tuning Hyper-parameters)
Model_Two = LogisticRegression(random_state = 31, class_weight = 'balanced',
                               max_iter = 500, penalty = 'l2', solver = 'newton-cg
')

# fit the model with data (Training Model)
Model_Two.fit(X_train,y_train)

# Prediction using model (test Model)
Model_Predictions = Model_Two.predict(X_test)
```

Model Three

Model Preparation

Hyper-Parameter Grid Search

```
#Calling Logistic Regression Model Object
Model_Three = LogisticRegression(random_state = 31, max_iter = 500)

#Tune Grid for C parameter
tune_grid = {
 'C' : np.arange(0.01, 0.99, 0.01)
}
tune_grid

#Setting up 10 Fold - Cross validation process
cv = RepeatedKFold(n_splits=10, n_repeats=5, random_state = 31)

#Implementing Grid Search
opt = GridSearchCV(
Model_Three, tune_grid, scoring='f1',
cv=cv, n_jobs=-1)
opt_results = opt.fit(X_train, y_train['Heart Disease'])
```

```
#F1 Scoring Metric
format_string = 'Average cross-validated in-sample F1 score {:.3f} {}'print(format_string.format(opt_results.best_score_,str(opt_results.best_params_)))
```

Optimal Grid Search Model

```
# Build a Logisitc Regression Model (Tuning Hyper-parameters)
Model_Three = LogisticRegression(random_state = 31, C = opt_results.best_params_['C'], max_iter = 500, solver='liblinear')

# fit the model with data (Training Model)
Model_Three.fit(X_train,y_train)

# Prediction using model (test Model)
Model_Predictions = Model_Three.predict(X_test)
```

Model Four

Model Preparation

Hyper-Parameter Grid Search

```
#LR object
Model_Four = LogisticRegression(random_state = 31, max_iter = 700)

#Logisict Regression Optimal Parameter Search Settings

LRparameter_grid = {

   'C' : [0.001, 0.01, 0.1, 1, 10, 100, 150, 200],

   'penalty' : ['l1','l2'],

   'max_iter' : list(range(100,800,1000)),

   'solver' : ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']

}

#Logisitc Regression Optimal Parameter Grid Search
LR_search = GridSearchCV(Model_Four, param_grid=LRparameter_grid, refit = True, verbose= 3, cv=5)

# fitting the model for grid search
LR_search.fit(X_train , y_train)
LR_search.best_params_

# summarize
print('Mean Accuracy: %.3f' % LR_search.best_score_)
print('Config: %s' % LR_search.best_params_)
```

Optimal Grid Search Model

```
# Build a Logisitc Regression Model (Tuning Hyper-parameters)
Model_Four = LogisticRegression(random_state = 31, C = 0.1, max_iter = 100, solver='liblinear')

# fit the model with data (Training Model)
Model_Four.fit(X_train,y_train)
```

# Prediction using model (test Model)
Model_Predictions = Model_Four.predict(X_test)

Model Five

Model Preparations

```
#Initiate minxmax Scaler Function
scalermm = MinMaxScaler()
data_mm = scalermm.fit_transform(scaling_df)
```

Building Model

```
# Build a Logisitc Regression Model (Tuning Hyper-parameters)
Model_Two = LogisticRegression(random_state = 31, class_weight = 'balanced',
                               max_iter = 500, penalty = 'l2', solver = 'newton-cg
')

# fit the model with data (Training Model)
Model_Two.fit(X_train,y_train)

# Prediction using model (test Model)
Model_Predictions = Model_Two.predict(X_test)
```

Model Six

Model Preparations

```
# Deleting features displaying strong co-occurrence
data.drop(['Marriage Status'], axis = 1, inplace = True)
data.drop(['Work Type'], axis = 1, inplace = True)
```

Building Model

```
# Build a Logisitc Regression Model (Tuning Hyper-parameters)
Model_Two = LogisticRegression(random_state = 31, class_weight = 'balanced',
                               max_iter = 500, penalty = 'l2', solver = 'newton-cg
')

# fit the model with data (Training Model)
Model_Two.fit(X_train,y_train)

# Prediction using model (test Model)
Model_Predictions = Model_Two.predict(X_test)
```