MuhammadHassanShah_20P-0025_C_Lab06

March 22, 2023

[1]:
```
import numpy as np
import pandas as pd
from sklearn import preprocessing

df = pd.read_csv("titanic_train.csv")

# df.head()
print(df.isna().sum())

# mean_age = df['age'].mean()
# mean_age


# label_encoder = preprocessing.LabelEncoder()
# for column in data:
#     data[column] = label_encoder.fit_transform(data[column])

# data.head()
# data = preprocessing.StandardScaler().fit(data)

# data.head()
```

```
passenger_id      0
pclass            0
name              0
sex               0
age             174
sibsp             0
parch             0
ticket            0
fare              1
cabin           659
embarked          1
boat            542
body            777
home.dest       386
survived          0
dtype: int64
```

[2]:
```
condition = df['body'].notnull()
d2 = df[condition]
d2.head()
```

[2]:
```
     passenger_id  pclass                                name     sex    age  sibsp  \
5            1083       3             Olsen, Mr. Henry Margido    male   28.0      0
14            602       3          Abbott, Mr. Rossmore Edward    male   16.0      1
29            162       1       Holverson, Mr. Alexander Oskar    male   42.0      1
40           1169       3         Saether, Mr. Simon Sivertsen    male   38.5      0
45           1239       3         Theobald, Mr. Thomas Leonard    male   34.0      0

    parch              ticket     fare cabin embarked boat   body  \
5       0              C 4001   22.525   NaN        S  NaN  173.0
14      1           C.A. 2673   20.250   NaN        S  NaN  190.0
29      0              113789   52.000   NaN        S  NaN   38.0
40      0  SOTON/O.Q. 3101262    7.250   NaN        S  NaN   32.0
45      0              363294    8.050   NaN        S  NaN  176.0

              home.dest  survived
5                   NaN         0
14  East Providence, RI         0
29         New York, NY         0
40                  NaN         0
45                  NaN         0
```

[3]:
```
d2[d2['survived'] == 1].head()
```

[3]:
```
Empty DataFrame
Columns: [passenger_id, pclass, name, sex, age, sibsp, parch, ticket, fare,
cabin, embarked, boat, body, home.dest, survived]
Index: []
```

1 We can see if a body has some value, it means that the passenger did not survive. Making it Binary attribute

[4]:
```
df['body'].fillna(1,inplace=True)
df['body'] = df['body'].replace(np.nan,0)
df.head()
```

[4]:
```
   passenger_id  pclass                                               name  \
0          1216       3                                 Smyth, Miss. Julia
1           699       3                                    Cacic, Mr. Luka
2          1267       3  Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go…
3           449       2              Hocking, Mrs. Elizabeth (Eliza Needs)
4           576       2                                    Veal, Mr. James
```

```
      sex   age  sibsp  parch  ticket     fare cabin embarked boat  body  \
0  female   NaN      0      0  335432   7.7333   NaN        Q   13   1.0
1    male  38.0      0      0  315089   8.6625   NaN        S  NaN   1.0
2  female  30.0      1      1  345773  24.1500   NaN        S  NaN   1.0
3  female  54.0      1      3   29105  23.0000   NaN        S    4   1.0
4    male  40.0      0      0   28221  13.0000   NaN        S  NaN   1.0

                  home.dest  survived
0                       NaN         1
1                   Croatia         0
2                       NaN         0
3      Cornwall / Akron, OH         1
4  Barre, Co Washington, VT         0
```

[5]:
```
condition = df['boat'].notnull()
d3 = df[condition]
d3.head(len(d3))
```

[5]:
```
     passenger_id  pclass                                               name  \
0            1216       3                                Smyth, Miss. Julia
3             449       2            Hocking, Mrs. Elizabeth (Eliza Needs)
7             560       2                              Sinkkonen, Miss. Anna
8            1079       3                                 Ohman, Miss. Velin
11             43       1  Bucknell, Mrs. William Robert (Emma Eliza Ward)
..            …     …                                                  …
840          1256       3                    Touma, Master. Georges Youssef
841           208       1  Minahan, Mrs. William Edward (Lillian E Thorpe)
842           709       3                          Carr, Miss. Helen "Ellen"
844           165       1                       Hoyt, Mr. Frederick Maxfield
847           467       2               Kantor, Mrs. Sinai (Miriam Sternin)

        sex   age  sibsp  parch  ticket     fare cabin embarked boat  body  \
0    female   NaN      0      0  335432   7.7333   NaN        Q   13   1.0
3    female  54.0      1      3   29105  23.0000   NaN        S    4   1.0
7    female  30.0      0      0  250648  13.0000   NaN        S   10   1.0
8    female  22.0      0      0  347085   7.7750   NaN        S    C   1.0
11   female  60.0      0      0   11813  76.2917   D15        C    8   1.0
..      …   …    …    …      …       …   …       …  …   …
840    male   7.0      1      1    2650  15.2458   NaN        C    C   1.0
841  female  37.0      1      0   19928  90.0000   C78        Q   14   1.0
842  female  16.0      0      0  367231   7.7500   NaN        Q   16   1.0
844    male  38.0      1      0   19943  90.0000   C93        S    D   1.0
847  female  24.0      1      0  244367  26.0000   NaN        S   12   1.0

                           home.dest  survived
0                                NaN         1
3              Cornwall / Akron, OH         1
```

```
7             Finland / Washington, DC         1
8                                  NaN         1
11                    Philadelphia, PA         1
..                                  …       …
840                                NaN         1
841                     Fond du Lac, WI         1
842  Co Longford, Ireland New York, NY         1
844        New York, NY /  Stamford CT         1
847                 Moscow / Bronx, NY         1

[308 rows x 15 columns]
```

[6]:
```
d3[d3['survived'] == 0].head(len(d3))
```

[6]:
```
     passenger_id  pclass                                               name  \
217            19       1                            Beattie, Mr. Thomson
219          1299       3                             Yasbeck, Mr. Antoni
272           968       3                  Lindell, Mr. Edvard Bengtsson
333           969       3  Lindell, Mrs. Edvard Bengtsson (Elin Gerda Per…
415           166       1                      Hoyt, Mr. William Fisher
443           544       2                       Renouf, Mr. Peter Henry
486           655       3                 Backstrom, Mr. Karl Alfred
644           921       3                           Keefe, Mr. Arthur

        sex   age  sibsp  parch    ticket     fare cabin embarked boat  body  \
217    male  36.0      0      0     13050  75.2417    C6        C    A   1.0
219    male  27.0      1      0      2659  14.4542   NaN        C    C   1.0
272    male  36.0      1      0    349910  15.5500   NaN        S    A   1.0
333  female  30.0      1      0    349910  15.5500   NaN        S    A   1.0
415    male   NaN      0      0  PC 17600  30.6958   NaN        C   14   1.0
443    male  34.0      1      0     31027  21.0000   NaN        S   12   1.0
486    male  32.0      1      0   3101278  15.8500   NaN        S    D   1.0
644    male   NaN      0      0    323592   7.2500   NaN        S    A   1.0

                              home.dest  survived
217                     Winnipeg, MN         0
219                              NaN         0
272                              NaN         0
333                              NaN         0
415                     New York, NY         0
443                    Elizabeth, NJ         0
486  Ruotsinphytaa, Finland New York, NY         0
644                              NaN         0
```

2 So If a boat was assigned, we still cannnot be sure if the passenger survived. But maximum passengers did.

[7]:
```
df['boat'] = df['boat'].replace(np.nan,'0')
df.head()
```

[7]:
```
   passenger_id  pclass                                               name  \
0          1216       3                                Smyth, Miss. Julia
1           699       3                                   Cacic, Mr. Luka
2          1267       3  Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go…
3           449       2              Hocking, Mrs. Elizabeth (Eliza Needs)
4           576       2                                   Veal, Mr. James

      sex   age  sibsp  parch  ticket     fare cabin embarked boat  body  \
0  female   NaN      0      0  335432   7.7333   NaN        Q   13   1.0
1    male  38.0      0      0  315089   8.6625   NaN        S    0   1.0
2  female  30.0      1      1  345773  24.1500   NaN        S    0   1.0
3  female  54.0      1      3   29105  23.0000   NaN        S    4   1.0
4    male  40.0      0      0   28221  13.0000   NaN        S    0   1.0

                  home.dest  survived
0                       NaN         1
1                   Croatia         0
2                       NaN         0
3      Cornwall / Akron, OH         1
4  Barre, Co Washington, VT         0
```

3 Home destination, passenger_id, p_class is irrelavant, so we drop it

[8]:
```
df.drop('home.dest',axis = 1,inplace = True)
df.drop('passenger_id',axis = 1,inplace = True)
df.drop('name',axis = 1,inplace = True)
df.head()
```

[8]:
```
   pclass     sex   age  sibsp  parch  ticket     fare cabin embarked boat  \
0       3  female   NaN      0      0  335432   7.7333   NaN        Q   13
1       3    male  38.0      0      0  315089   8.6625   NaN        S    0
2       3  female  30.0      1      1  345773  24.1500   NaN        S    0
3       2  female  54.0      1      3   29105  23.0000   NaN        S    4
4       2    male  40.0      0      0   28221  13.0000   NaN        S    0

   body  survived
0   1.0         1
1   1.0         0
```

```
2   1.0         0
3   1.0         1
4   1.0         0
```

[9]:
```
print(df.isna().sum())
```

```
pclass        0
sex           0
age         174
sibsp         0
parch         0
ticket        0
fare          1
cabin       659
embarked      1
boat          0
body          0
survived      0
dtype: int64
```

[10]:
```
df['cabin'] = df['cabin'].replace(np.nan,'0')
df['embarked'] = df['embarked'].replace(np.nan,'0')
df['fare'] = df['fare'].replace(np.nan,0)
df.head()
```

[10]:

	pclass	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	survived
0	3	female	NaN	0	0	335432	7.7333	0	Q	13	1.0	1
1	3	male	38.0	0	0	315089	8.6625	0	S	0	1.0	0
2	3	female	30.0	1	1	345773	24.1500	0	S	0	1.0	0
3	2	female	54.0	1	3	29105	23.0000	0	S	4	1.0	1
4	2	male	40.0	0	0	28221	13.0000	0	S	0	1.0	0

[11]:
```
df['age'] = df['age'].replace(np.nan,df['age'].mean())
df.head()
```

[11]:

	pclass	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	survived
0	3	female	29.519847	0	0	335432	7.7333	0	Q	13	1.0	1
1	3	male	38.000000	0	0	315089	8.6625	0	S	0	1.0	0
2	3	female	30.000000	1	1	345773	24.1500	0	S	0	1.0	0
3	2	female	54.000000	1	3	29105	23.0000	0	S	4	1.0	1

```
4       2    male  40.000000      0      0   28221  13.0000     0        S

  boat  body  survived
0   13   1.0         1
1    0   1.0         0
2    0   1.0         0
3    4   1.0         1
4    0   1.0         0
```

[12]:
```
print(df.isna().sum())
```

```
pclass      0
sex         0
age         0
sibsp       0
parch       0
ticket      0
fare        0
cabin       0
embarked    0
boat        0
body        0
survived    0
dtype: int64
```

[13]:
```
import matplotlib.pyplot as mp
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score

y = df['survived']
df.drop('survived',axis = 1,inplace = True)
X_test, X_train, y_test, y_train = train_test_split(df,y,test_size=0.2,␣
 ↪random_state=42, shuffle=True)

# X_train.hist(bins="auto",figsize=(9,7),grid=False)
df.head()
```

[13]:
```
   pclass     sex        age  sibsp  parch  ticket     fare cabin embarked  \
0       3  female  29.519847      0      0  335432   7.7333     0        Q
1       3    male  38.000000      0      0  315089   8.6625     0        S
2       3  female  30.000000      1      1  345773  24.1500     0        S
3       2  female  54.000000      1      3   29105  23.0000     0        S
4       2    male  40.000000      0      0   28221  13.0000     0        S

  boat  body
```

```
0   13   1.0
1    0   1.0
2    0   1.0
3    4   1.0
4    0   1.0
```

[20]:
```
label_encoder = preprocessing.LabelEncoder()
for column in df:
    df[column] = label_encoder.fit_transform(df[column])
```

[21]:
```
df.head(len(df))
```

[21]:

	pclass	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body
0	2	0	40	0	0	283	25	0	2	5	0
1	2	1	51	0	0	257	48	0	3	0	0
2	2	0	41	1	1	307	124	0	3	0	0
3	1	0	70	1	3	237	120	0	3	14	0
4	1	1	54	0	0	228	77	0	3	0	0
..	…	…	…	…	…	…	…	…	…	…	…
845	0	1	71	0	0	488	172	62	3	0	0
846	0	1	75	0	0	58	143	23	1	0	54
847	1	0	32	1	0	145	130	0	3	4	0
848	2	0	8	1	1	630	80	0	3	0	0
849	1	1	68	0	0	160	77	0	3	0	6

[850 rows x 11 columns]

[22]:
```
print(df.dtypes)
```

```
pclass      int64
sex         int64
age         int64
sibsp       int64
parch       int64
ticket      int64
fare        int64
cabin       int64
embarked    int64
boat        int64
body        int64
dtype: object
```

[23]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
  ↪(3,3),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("3,3 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_3169/1023893291.py in <module>
      1 mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(3,3),random_state = 1)
----> 2 mlp.fit(X_train,y_train)
      3 y_pred = mlp.predict(X_test)
      4 y_score = mlp.score(X_test,y_test)
      5 accuracy = accuracy_score(y_test, y_pred)

~/anaconda3/lib/python3.9/site-packages/sklearn/neural_network/
 ↪_multilayer_perceptron.py in fit(self, X, y)
    750             Returns a trained MLP model.
    751         """
--> 752         return self._fit(X, y, incremental=False)
    753
    754     def _check_solver(self):

~/anaconda3/lib/python3.9/site-packages/sklearn/neural_network/
 ↪_multilayer_perceptron.py in _fit(self, X, y, incremental)
    391         )
    392
--> 393         X, y = self._validate_input(X, y, incremental, reset=first_pass
    394
    395         n_samples, n_features = X.shape

~/anaconda3/lib/python3.9/site-packages/sklearn/neural_network/
 ↪_multilayer_perceptron.py in _validate_input(self, X, y, incremental, reset)
   1098
   1099     def _validate_input(self, X, y, incremental, reset):
-> 1100         X, y = self._validate_data(
   1101             X,
   1102             y,

~/anaconda3/lib/python3.9/site-packages/sklearn/base.py in _validate_data(self,␣
 ↪X, y, reset, validate_separately, **check_params)
```

```
    579                 y = check_array(y, **check_y_params)
    580             else:
--> 581                 X, y = check_X_y(X, y, **check_params)
    582             out = X, y
    583

~/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in␣
 ↪check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples,␣
 ↪ensure_min_features, y_numeric, estimator)
    962         raise ValueError("y cannot be None")
    963
--> 964     X = check_array(
    965         X,
    966         accept_sparse=accept_sparse,

~/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in␣
 ↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
 ↪ensure_min_features, estimator)
    744                     array = array.astype(dtype, casting="unsafe",␣
 ↪copy=False)
    745                 else:
--> 746                     array = np.asarray(array, order=order, dtype=dtype)
    747             except ComplexWarning as complex_warning:
    748                 raise ValueError(

~/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in␣
 ↪__array__(self, dtype)
   2062
   2063     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.
 ↪ndarray:
-> 2064         return np.asarray(self._values, dtype=dtype)
   2065
   2066     def __array_wrap__(

ValueError: could not convert string to float: 'male'
```

4 For some reasons, it gives this behaviour. I restarted the kernel and it becomes fine. But when I do it all over again it does the same thing. I'll take (3,4) as initial model and continue on.

5 Even tho all the datatypes are int, it still does this. IDK why

[18]:
```
print(df.dtypes)
```

```
pclass      int64
sex         int64
age         int64
sibsp       int64
parch       int64
ticket      int64
fare        int64
cabin       int64
embarked    int64
boat        int64
body        int64
dtype: object
```

[29]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(3,4),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("3,4 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
3,4 neurons
Accuracy: 0.6529411764705882
Precision: 0.5248618784530387
Recall: 0.3877551020408163
F1 score: 0.4460093896713615
```

[ ]:
```
```

[30]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(4,4),random_state = 1)
```

```
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("4,4 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
4,4 neurons
Accuracy: 0.6397058823529411
Precision: 0.0
Recall: 0.0
F1 score: 0.0
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[32]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
↪(4,5),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("4,5 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
4,5 neurons
Accuracy: 0.6397058823529411
Precision: 0.0
Recall: 0.0
F1 score: 0.0
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[34]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(5,5),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("5,5 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
5,5 neurons
Accuracy: 0.6720588235294118
Precision: 0.6122448979591837
Recall: 0.24489795918367346
F1 score: 0.3498542274052478
```

[35]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(6,6),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("6,6 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
6,6 neurons
Accuracy: 0.9720588235294118
Precision: 0.9669421487603306
Recall: 0.9551020408163265
F1 score: 0.9609856262833676
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:549:
```

```
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

[36]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(7,7),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("7,7 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
7,7 neurons
Accuracy: 0.9705882352941176
Precision: 0.9591836734693877
Recall: 0.9591836734693877
F1 score: 0.9591836734693877

/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:549:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

[37]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(8,8),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("8,8 neurons")
print('Accuracy:', accuracy)
```

```
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
8,8 neurons
Accuracy: 0.9705882352941176
Precision: 0.9629629629629629
Recall: 0.9551020408163265
F1 score: 0.959016393442623
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:549:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

[39]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(10,10),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("10,10 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
10,10 neurons
Accuracy: 0.9485294117647058
Precision: 0.926829268292683
Recall: 0.9306122448979591
F1 score: 0.9287169042769857
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:549:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

[40]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(9,9),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("4,4 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
4,4 neurons
Accuracy: 0.7573529411764706
Precision: 0.8225806451612904
Recall: 0.4163265306122449
F1 score: 0.5528455284552846
```

6 We are getting maximum accuracy at (8,8)

[41]:
```
mlp = MLPClassifier(solver="lbfgs",alpha = 1e-5,hidden_layer_sizes =␣
 ↪(8,8),random_state = 1)
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
y_score = mlp.score(X_test,y_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("8,8 neurons")
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
8,8 neurons
Accuracy: 0.9705882352941176
Precision: 0.9629629629629629
Recall: 0.9551020408163265
F1 score: 0.959016393442623
```

```
/home/h/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:549:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

7 Initial = (3,4)

8 Fine Tuned = (8,8)

[26]:
```
in_Accuracy =  0.6529411764705882
in_Precision = 0.5248618784530387
in_Recall = 0.3877551020408163
in_F1_score = 0.4460093896713615
f_Accuracy= 0.9705882352941176
f_Precision= 0.9629629629629629
f_Recall= 0.9551020408163265
f_F1_score= 0.959016393442623
data = {'Accuracy': [in_Accuracy,f_Accuracy],
        'Precision': [in_Precision,f_Precision],
        'Recall':[in_Recall,f_Recall],
        'F1_score': [in_F1_score,f_F1_score]
       }
data = pd.DataFrame(data,index=['Initial','Final'])
data.head()
```

[26]:

	Accuracy	Precision	Recall	F1_score
Initial	0.652941	0.524862	0.387755	0.446009
Final	0.970588	0.962963	0.955102	0.959016

The initial model have .65 accuracy which is very low, after hit and trial using different layers, we can see improvement. But it doesn't always means the bigger the better. I did, and (9.9) and (10,10), we were decreasing in accuracy. So w