# ANN

In [160]:

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.50,random\_state**=**0) print(X\_train.shape)

print(X\_test.shape) print(y\_train.shape) print(y\_test.shape)

(138, 15)

(138, 15)

(138,)

(138,)

In [161]:

**from** sklearn.preprocessing **import** StandardScaler scaler**=**StandardScaler()

X\_train**=**scaler.fit\_transform(X\_train) X\_test**=**scaler.transform(X\_test)

In [162]:

**import** warnings

warnings.filterwarnings('ignore')

**from** sklearn.neural\_network **import** MLPRegressor

ann\_model**=**MLPRegressor(hidden\_layer\_sizes**=**(128,64,32),activation**=**'relu',solver**=**'lbfgs') ann\_model.fit(X\_train,y\_train)

Out[162]:

▾

MLPRegressor

MLPRegressor(hidden\_layer\_sizes=(128, 64, 32), solver='lbfgs')

In [163]:

y\_pred**=**ann\_model.predict(X\_test) y\_pred

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[163]: | array([-0.27046645, | 1.11420306, | 1.08119798, | 0.99855865, | 0.85089268, |
|  | 1.03020246, | 1.10064246, | 0.99658885, | 1.03396644, | -0.35122544, |
|  | 0.9355521 , | 1.0812097 , | 1.02217053, | 1.00651045, | 0.60816984, |
|  | 0.72060743, | 1.09897109, | 0.09602785, | 0.93502529, | -0.1795292 , |
|  | 1.00088269, | 1.12379087, | 1.03445763, | 1.11736311, | -0.02374211, |
|  | 1.12034558, | 1.05507871, | 0.99283378, | 0.98878657, | 0.99784774, |
|  | 0.93312314, | 0.97423255, | 1.08226353, | 1.00558863, | 1.00996518, |
|  | 0.74354272, | -0.04148026, | 1.11505494, | 0.76466434, | 0.99053938, |
|  | 1.04484394, | 0.95998932, | 0.74473401, | 0.53927351, | 0.96967064, |
|  | 1.12121031, | -0.12568266, | 0.88086758, | 0.58331154, | 0.99763951, |
|  | 0.9925576 , | -0.27234935, | 0.89640104, | 0.52923437, | 1.01365338, |
|  | 0.08252559, | 1.0415732 , | 1.12555965, | 0.00620682, | 0.34468117, |
|  | 1.012124 , | 0.98080112, | 1.00419668, | 1.01818306, | 0.99279325, |
|  | 0.99407564, | 0.99415995, | 0.95270087, | 0.52614255, | 0.97924742, |
|  | 0.0252738 , | 0.38481667, | 0.90413418, | 0.44521408, | 0.98998584, |
|  | 0.55718558, | 1.05580083, | 0.99225173, | 1.00142334, | 1.02567507, |
|  | -0.03524893, | 0.10791324, | 1.02547129, | 0.9518621 , | 0.88645931, |
|  | 0.01715826, | 0.32812365, | 1.02467144, | 1.06840844, | 0.9903359 , |
|  | 1.38624109, | 0.9553072 , | 1.20999602, | 1.00292861, | 0.89077127, |
|  | 1.24003028, | 0.99587639, | 0.99408538, | -0.02719458, | -0.03850283, |
|  | 1.01230475, | 1.00774796, | 0.00277839, | 1.05886746, | 1.07943519, |
|  | 1.07667188, | 0.90798554, | 0.74995355, | 1.03025105, | -0.12231914, |
|  | 0.99654681, | 0.9947503 , | 1.07403895, | 0.98672129, | 0.83938453, |
|  | 1.04902887, | 0.9823371 , | 0.50050329, | 0.55416735, | 1.04616607, |
|  | 0.01509583, | 0.97688152, | 0.68568851, | 0.9523753 , | 1.02306403, |
|  | 0.81576317, | 1.00948095, | 0.84185211, | -0.03077194, | 0.92916801, |
|  | 0.10791324, | 1.01271946, | -0.08388515, | 1.25519857, | 0.63832129, |
|  | 1.00640523, | 1.04678753, | 0.40416379]) |  |  |

In [164]:

**from** sklearn.metrics **import** r2\_score r2\_score**=**r2\_score(y\_test,y\_pred)

print('R-square score:',r2\_score)

R-square score: 0.09410233952613589

In [165]:

train\_accuracy**=**ann\_model.score(X\_train,y\_train) print('train\_accuracy:',train\_accuracy)

test\_accuracy**=**ann\_model.score(X\_test,y\_test) print('test\_accuracy:',test\_accuracy)

train\_accuracy: 0.9998859603548554

test\_accuracy: 0.94102339526135891

In [166]:

**import** math

**from** sklearn **import** metrics

print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,y\_pred)) print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,y\_pred))

print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

Mean Absolute Error: 0.1785092389238528 Mean Squared Error: 0.11226204992219699

Root Mean Squared Error: 0.335055293828044

In [169]:

**import** matplotlib.pyplot **as** plt

logistic\_regression\_accuracy **=**0.9464285714285714

random\_forest\_accuracy **=**0.9036144578313254 ANN\_model**=**0.9998859603548554

accuracy\_scores **=** [linear\_regression\_accuracy, random\_forest\_accuracy,ANN\_model] model\_names **=** ['Linear Regression', 'Random Forest Regression','ANN\_model']

plt.bar(model\_names, accuracy\_scores) plt.xlabel('Regression Models')

plt.ylabel('Test Accuracy')

plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression vs ANN\_model')

plt.show()

