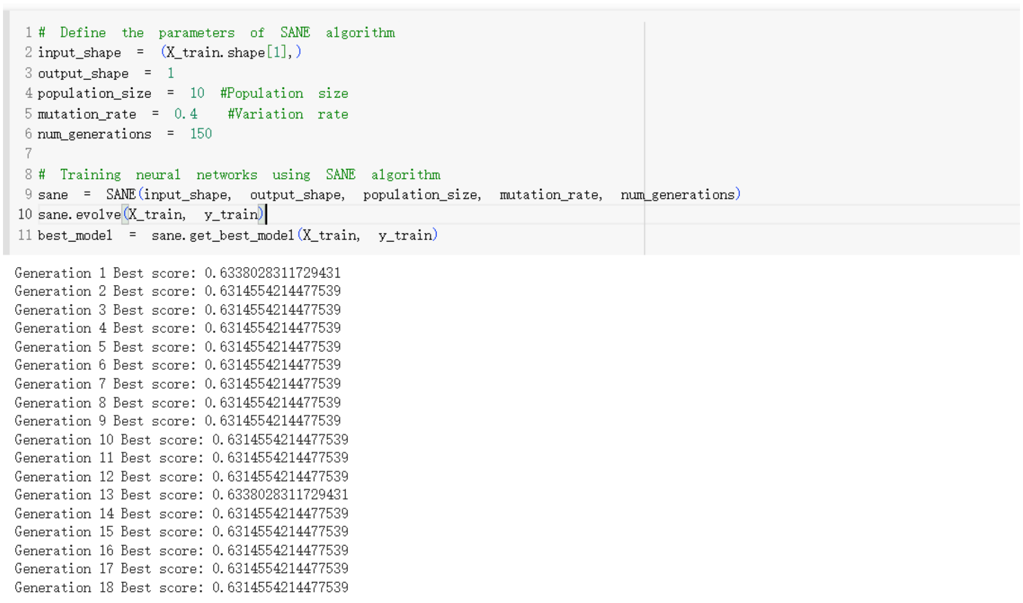
```
# Define the parameters of SANE algorithm
input_shape = (X_train.shape[1],)
output_shape = 1
population_size = 10 #Population size
mutation_rate = 0.4   #Variation rate
num_generations = 150

# Training neural networks using SANE algorithm
sane = SANE(input_shape, output_shape, population_size, mutation_rate, num_generations)
sane.evolve(X_train, y_train)
best_model = sane.get_best_model(X_train, y_train)
```



```
# Define the parameters of SANE algorithm
input_shape = (X_train.shape[1],)
output_shape = 1
population_size = 10 #Population size
mutation_rate = 0.4   #Variation rate
num_generations = 150

# Training neural networks using SANE algorithm
sane = SANE(input_shape, output_shape, population_size, mutation_rate, num_generations)
sane.evolve(X_train, y_train)
best_model = sane.get_best_model(X_train, y_train)
```

```
Generation 1 Best score: 0.6338028311729431
Generation 2 Best score: 0.6314554214477539
Generation 3 Best score: 0.6314554214477539
Generation 4 Best score: 0.6314554214477539
Generation 5 Best score: 0.6314554214477539
Generation 6 Best score: 0.6314554214477539
Generation 7 Best score: 0.6314554214477539
Generation 8 Best score: 0.6314554214477539
Generation 9 Best score: 0.6314554214477539
Generation 10 Best score: 0.6314554214477539
Generation 11 Best score: 0.6314554214477539
Generation 12 Best score: 0.6314554214477539
Generation 13 Best score: 0.6338028311729431
Generation 14 Best score: 0.6314554214477539
Generation 15 Best score: 0.6314554214477539
Generation 16 Best score: 0.6314554214477539
Generation 17 Best score: 0.6314554214477539
Generation 18 Best score: 0.6314554214477539
```

Рисунок- 1

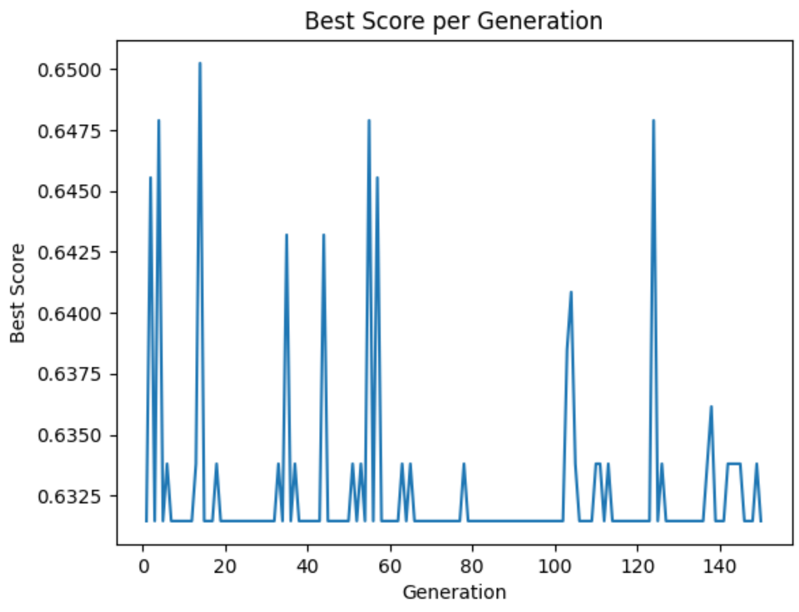


Рисунок -2

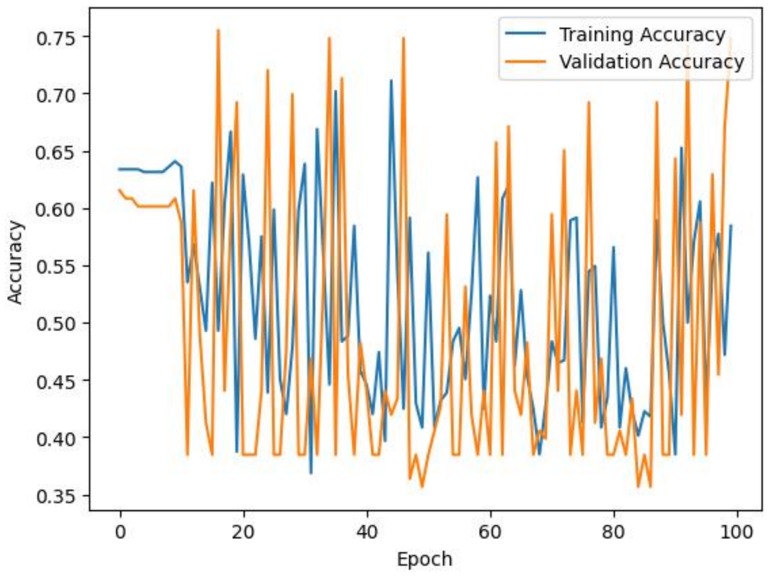
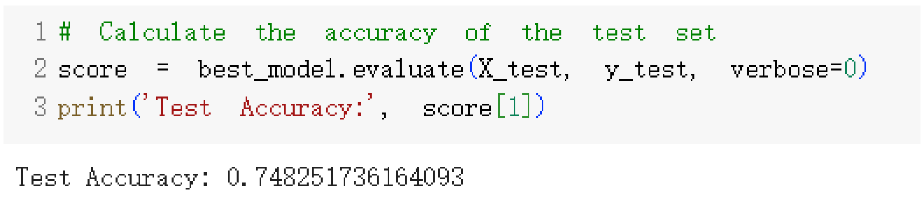


Рисунок 3



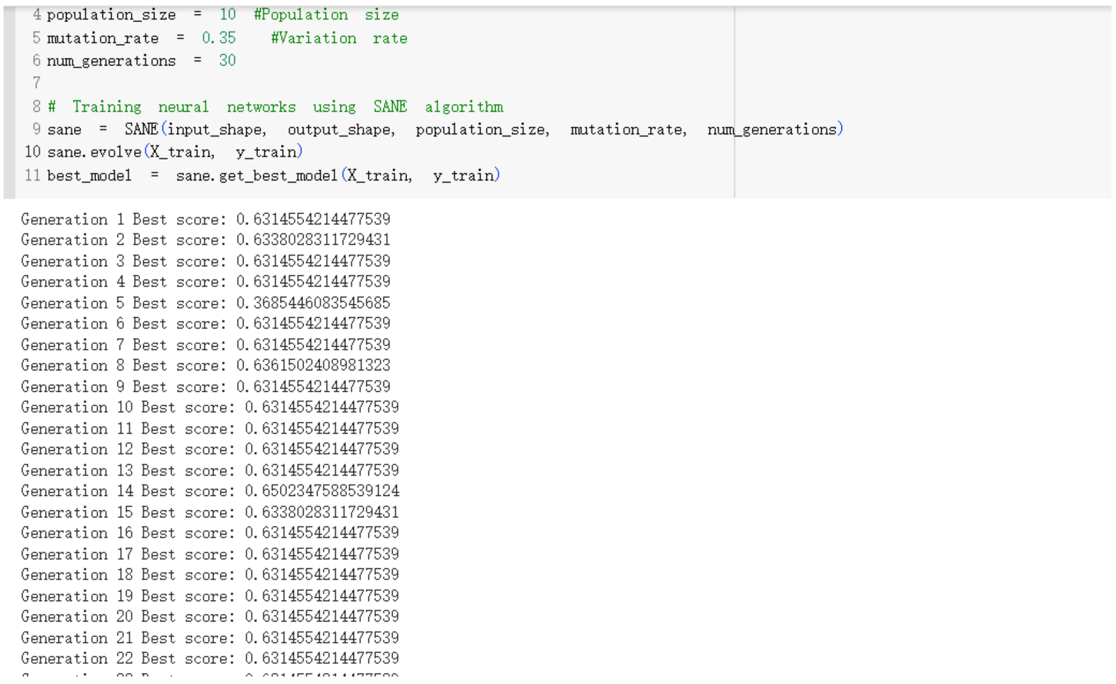
```
# Calculate the accuracy of the test set
score = best_model.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy:', score[1])
```

```
Test Accuracy: 0.748251736164093
```

Рисунок 4

На рисунках 1, 2, 3 и 4 показано количество "Generation" при "mutation_rate = 0.4", "num_generations = 150" и "Best score" при "mutation_rate = 0.4", "num_generations = 150" и "num_generations = 150", Generation " и "Best score" графики. Получение лучших оценок для каждого поколения, тестирование лучшей модели и построение графика сходимости.

```
# Define the parameters of SANE algorithm
input_shape = (X_train.shape[1],)
output_shape = 1
population_size = 10 #Population size
mutation_rate = 0.35   #Variation rate
num_generations = 30

# Training neural networks using SANE algorithm
sane = SANE(input_shape, output_shape, population_size, mutation_rate, num_generations)
sane.evolve(X_train, y_train)
best_model = sane.get_best_model(X_train, y_train)
```



```
population_size  =  10  #Population  size
mutation_rate  =  0.35    #Variation  rate
num_generations  =  30

#  Training  neural  networks  using  SANE  algorithm
sane  =  SANE(input_shape,  output_shape,  population_size,  mutation_rate,  num_generations)
sane.evolve(X_train,  y_train)
best_model  =  sane.get_best_model(X_train,  y_train)
```

```
Generation 1 Best score: 0.6314554214477539
Generation 2 Best score: 0.6338028311729431
Generation 3 Best score: 0.6314554214477539
Generation 4 Best score: 0.6314554214477539
Generation 5 Best score: 0.3685446083545685
Generation 6 Best score: 0.6314554214477539
Generation 7 Best score: 0.6314554214477539
Generation 8 Best score: 0.6361502408981323
Generation 9 Best score: 0.6314554214477539
Generation 10 Best score: 0.6314554214477539
Generation 11 Best score: 0.6314554214477539
Generation 12 Best score: 0.6314554214477539
Generation 13 Best score: 0.6314554214477539
Generation 14 Best score: 0.6502347588539124
Generation 15 Best score: 0.6338028311729431
Generation 16 Best score: 0.6314554214477539
Generation 17 Best score: 0.6314554214477539
Generation 18 Best score: 0.6314554214477539
Generation 19 Best score: 0.6314554214477539
Generation 20 Best score: 0.6314554214477539
Generation 21 Best score: 0.6314554214477539
Generation 22 Best score: 0.6314554214477539
```

Рисунок 5

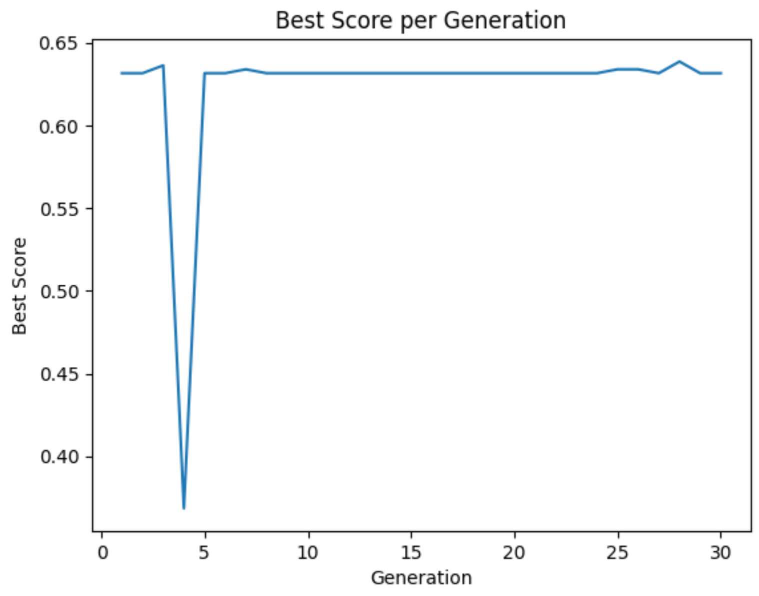


Рисунок 6

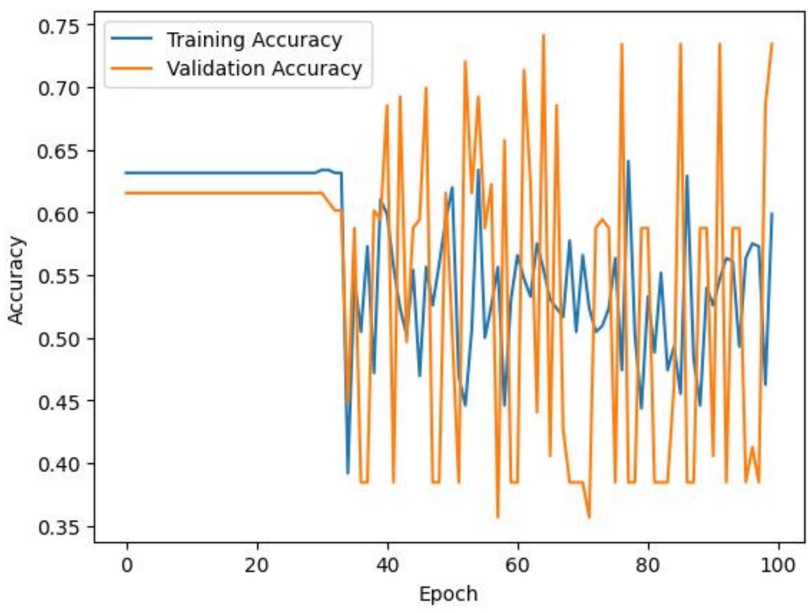
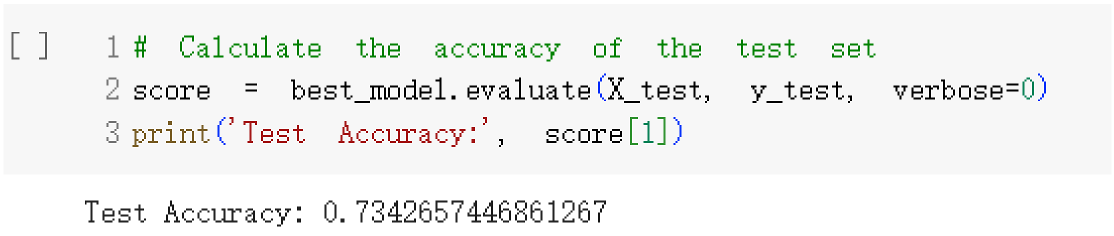


Рисунок 7

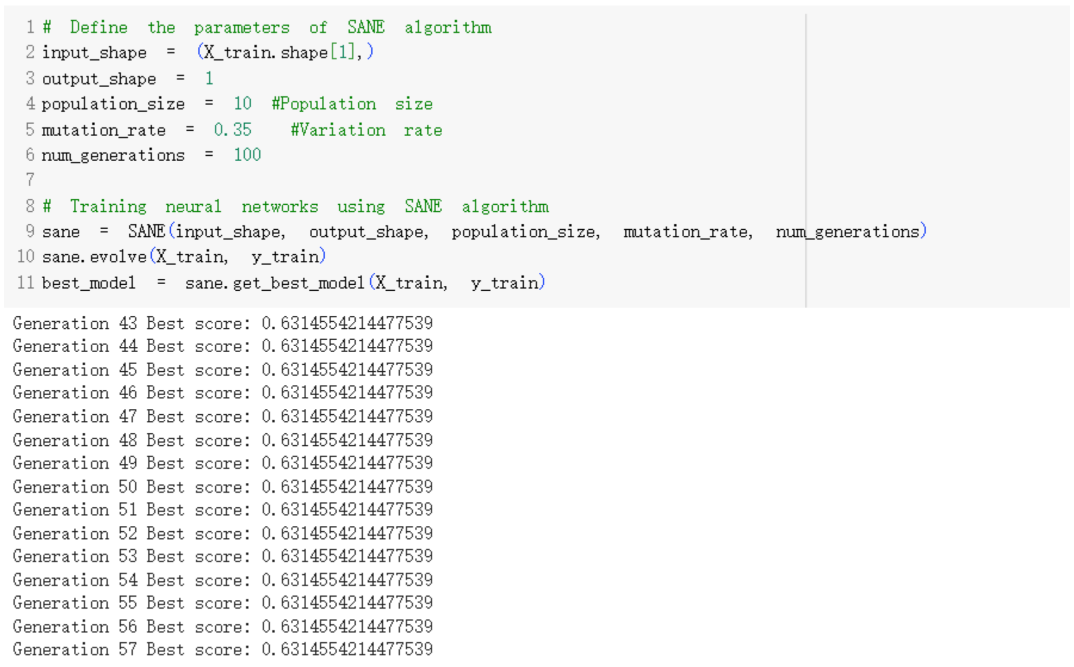


```
[ ]   # Calculate the accuracy of the test set
      score = best_model.evaluate(X_test, y_test, verbose=0)
      print('Test Accuracy:', score[1])

    Test Accuracy: 0.7342657446861267
```

Рисунок 8

На рисунках 5, 6, 7 и 8 показано количество "Generation" при "mutation_rate = 0.4", "num_generations = 150" и "Best score" при "mutation_rate = 0.4", "num_generations = 150" и "num_generations = 150", Generation " и "Best score" графики. Получение лучших оценок для каждого поколения, тестирование лучшей модели и построение графика сходимости.

```
# Define the parameters of SANE algorithm
input_shape = (X_train.shape[1],)
output_shape = 1
population_size = 10 #Population size
mutation_rate = 0.35   #Variation rate
num_generations = 100

# Training neural networks using SANE algorithm
sane = SANE(input_shape, output_shape, population_size, mutation_rate, num_generations)
sane.evolve(X_train, y_train)
best_model = sane.get_best_model(X_train, y_train)
```



```
# Define the parameters of SANE algorithm
input_shape = (X_train.shape[1],)
output_shape = 1
population_size = 10 #Population size
mutation_rate = 0.35   #Variation rate
num_generations = 100

# Training neural networks using SANE algorithm
sane = SANE(input_shape, output_shape, population_size, mutation_rate, num_generations)
sane.evolve(X_train, y_train)
best_model = sane.get_best_model(X_train, y_train)
```

```
Generation 43 Best score: 0.6314554214477539
Generation 44 Best score: 0.6314554214477539
Generation 45 Best score: 0.6314554214477539
Generation 46 Best score: 0.6314554214477539
Generation 47 Best score: 0.6314554214477539
Generation 48 Best score: 0.6314554214477539
Generation 49 Best score: 0.6314554214477539
Generation 50 Best score: 0.6314554214477539
Generation 51 Best score: 0.6314554214477539
Generation 52 Best score: 0.6314554214477539
Generation 53 Best score: 0.6314554214477539
Generation 54 Best score: 0.6314554214477539
Generation 55 Best score: 0.6314554214477539
Generation 56 Best score: 0.6314554214477539
Generation 57 Best score: 0.6314554214477539
```

Рисунок 9

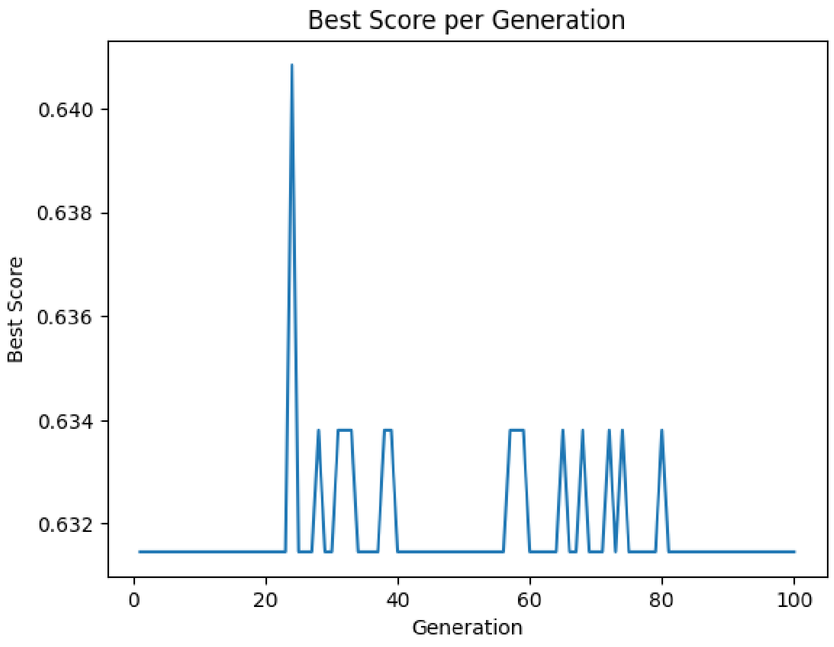


Рисунок 10

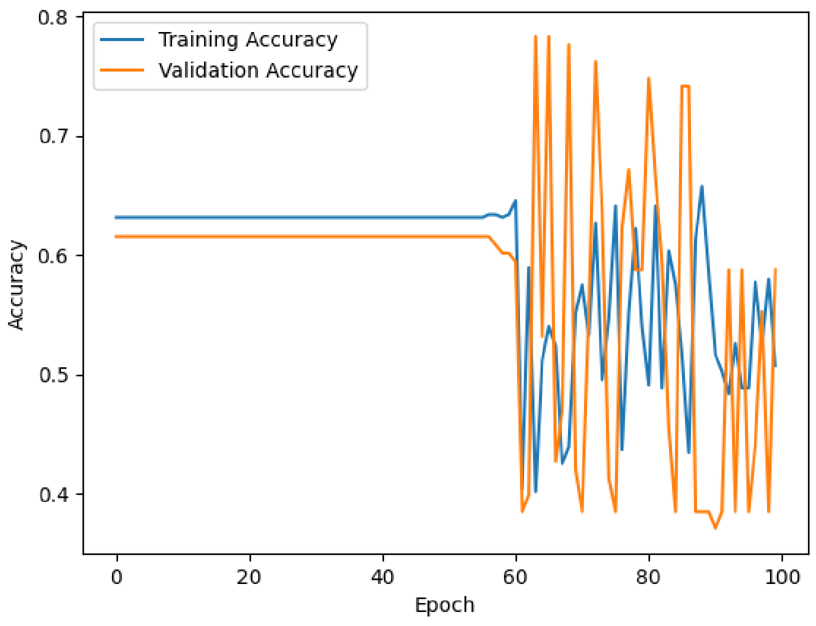


Рисунок 11

```
# Calculate the accuracy of the test set
score = best_model.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy:', score[1])
```

```
Test Accuracy: 0.7342657446861267
```

Рисунок 12

Table 1

mutation_rate,num_generations	Generation :	Best score:	Test Accuracy:
0.35 , 100	32	0.64	0.58
0.4 , 150	56	0.74	0.74
0.35 ,30	14	0.65	0.73

Table 1 shows the results of the SANE algorithm with different mutation coefficients and surrogate values. Each column is explained below:

Mutation rate: This column indicates the mutation rate used in the SANE algorithm. It indicates the probability of mutations occurring during reproduction.

Num Generations: This column indicates the number of generations performed by the SANE algorithm. It indicates the number of iterations or cycles of the algorithm.

Generations: This column indicates the number of generations of the SANE algorithm during evolution.

Best result: This column shows the best result achieved in each generation. It represents the highest fit or accuracy score obtained for any individual model in that generation.

Test Accuracy: This column shows the accuracy of the best model obtained in each generation when evaluated on the test set.

From this table we can observe the following:

For a mutation factor of 0.35 and 100 generations, the best score is 32, which corresponds to a test accuracy of 0.58.

For a mutation factor of 0.4 and 150 generations, the best result is 56 and the corresponding test accuracy is 0.74.

For a mutation factor of 0.35 and 30 generations, the best result is 14 points, and the corresponding test accuracy is 0.73.

Based on these results, we can conclude that a higher mutation rate (0.4) and a higher number of generations (150) results in better performance, as indicated by higher best score and test accuracy. However, it is important to note that these results cannot be definitive, as they are based on a limited number of experiments. It is recommended to perform several runs and consider other factors, such as computational resources and time constraints, to determine the best combination of mutation rate and number of generations for a given task.