Business Applications of A.I. (ISTM6214)

Project Title:

Breast Cancer Prediction and Classfication

Team Members:

Zhiguang Chen, Kaiyu Luo,

Long Xin, Yuhan Zhang_

Executive Summary

Among all types of illnesses that cause death in this world, cancer is the second leading cause of death worldwide. According to the report from the World Health Organization, cancer caused around 9.6 million in 2018, which means that there will be a death caused by cancer on every 6 deaths globally. Since there are so many different types of cancer, Lung and Breast cancer are the top two common cancers globally and each of them had 2.09 million cases reported to WHO. Instead of other lethal cancers, most of breast cancer can be cured through proper treatment if patients can be diagnosed at an early stage of breast cancer.

In this project, our team plans to apply numerals Machine Learning and Deep Learning models by using keras to improve the general accuracy of breast cancer diagnosis. The dataset we use in this project contains 569 real cases with unique ID numbers, and each of them comes with final diagnosis. And each case has real-valued features, which were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, of cell nucleus recorded to diagnose the nature of tumor. To accomplish our objective, we have decided to build four models, which are KNN, Random Forest, Logistic Regression, and ANN to reach the best performance.

Data Description

Acquired from Kaggle.com[1], this dataset contains 569 real cases of breast cancer and each case provided specific features of cell nucleus. Through this information we are able to diagnose whether the nature of the tumor is malignant or benign.

There are total 32 columns in this dataset, include id, diagnosis, radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean, radius_se, texture_se, perimeter_se, area_se, smoothness_se, compactness_se, concavity_se, concave points_se, symmetry_se, fractal_dimension_se, radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst, fractal_dimension_worst.

Our target variable is Diagnosis.

Below are descriptions of each column.

- id -------------------------- ID number

[1] https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

- diagnosis ----------------- the diagnosis of breast tissues (M = malignant, B = benign)
- radius_mean ------------- mean of distances from center to points on the perimeter
- texture_mean ------------ standard deviation of gray-scale values
- perimeter_mean --------- mean size of the core tumor
- area_mean
- smoothness_mean --------mean of local variation in radius lengths
- compactness_mean ------ mean of perimeter^2 / area - 1.0
- concavity_mean ----------mean of severity of concave portions of the contour
- concave points_mean --- mean for number of concave portions of the contour
- symmetry_mean
- fractal_dimension_mean-mean for "coastline approximation" - 1
- radius_se -----------------standard error for the mean of distances from center to points on the perimeter
- texture_se ----------------standard error for standard deviation of gray-scale values
- perimeter_se
- area_se
- smoothness_se ---------- standard error for local variation in radius lengths
- compactness_se --------- standard error for perimeter^2 / area - 1.0
- concavity_se ------------ standard error for severity of concave portions of the contour
- concave points_se --------standard error for number of concave portions of the contour
- symmetry_se
- fractal_dimension_se --- standard error for "coastline approximation" - 1
- radius_worst ------------"worst" or largest mean value for mean of distances from center to points on the perimeter
- texture_worst ----------- "worst" or largest mean value for standard deviation of gray-scale values
- perimeter_worst
- area_worst
- smoothness_worst ------ "worst" or largest mean value for local variation in radius lengths
- compactness_worst ------"worst" or largest mean value for perimeter^2 / area - 1.0
- concavity_worst ---------"worst" or largest mean value for severity of concave

portions of the contour

- concave points_worst ---"worst" or largest mean value for number of concave portions of the contour
- symmetry_worst
- fractal_dimension_worst --- "worst" or largest mean value for "coastline approximation" - 1

Research Questions

While the progress of the project moving forward, we spotted some challenging problems in this process and these all could impact our final results.

- Dataset Searching

The very first problem we were facing is to find the proper dataset that contains all the information we need to make the prediction. Improper dataset could lead the whole project to another direction.

- Data Cleaning

In the dataset we found, there were some variables that are non numerical, such as the diagnosis result of each case and we have to convert these values to have the model run through. Also, there are some irrelevant variables that existed in the original dataset, and to minimize the unnecessary impacts from these irrelevant variables, data cleaning is the very first step we did for before we start testing our models.

- Modeling

Since we are applying for ML and DL for this dataset, it is challenging to optimize our model to accomplish our objectives. In this situation, we decided to build different classification models to cross contrast the final result, and to test for the best parameters.

Methodology

- Data Cleaning:

The very first step for any well-down analysis is to perform the data cleaning to make sure the prediction is as accurate as possible.

- Data Visualization

In order to have a better acknowledgement of our dataset, we visualized the dataset by plotting out the frequency of cancer stage.

- Data Split:

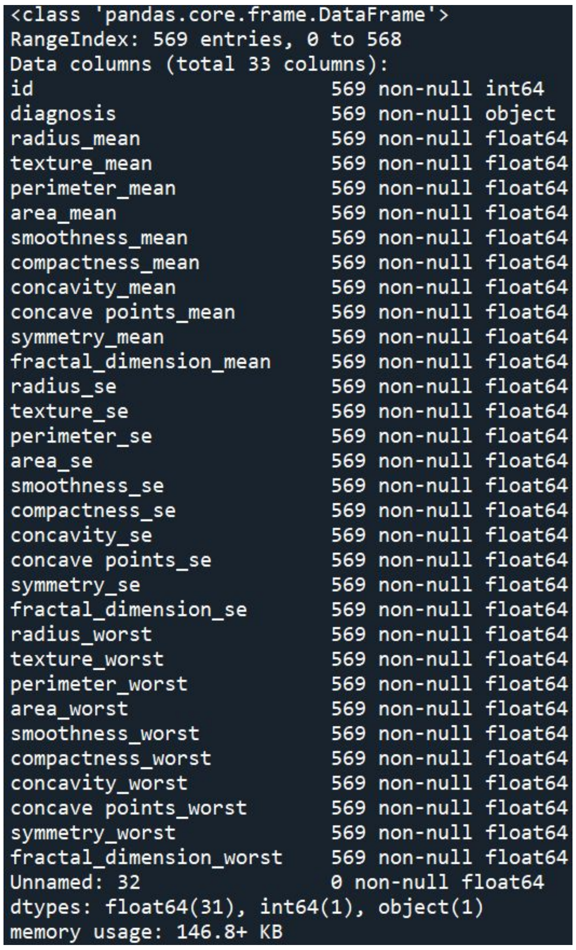
To achieve the best performance, we randomly split the dataset with 80% for training dataset and 20% for testing dataset.

- Modeling:

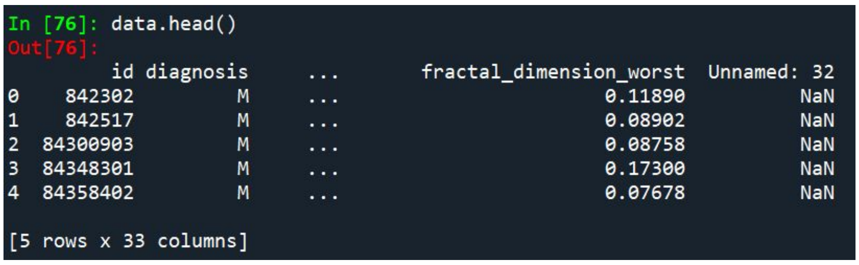
We decided to build KNN, RF, Logistic Regression and ANN models using keras. In this case, our prediction accuracy can be precise with diverse classification models.

Results and Findings

- Data cleaning



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

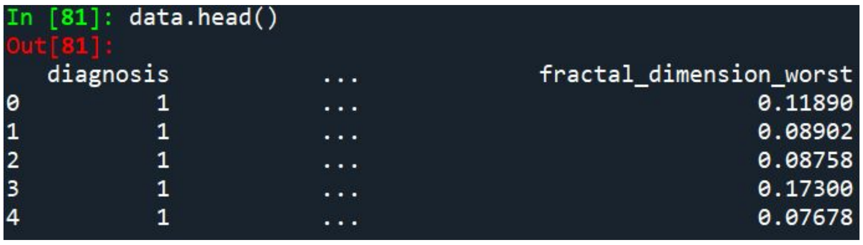


```
In [76]: data.head()
Out[76]:
         id diagnosis     ...       fractal_dimension_worst  Unnamed: 32
0    842302         M     ...                       0.11890          NaN
1    842517         M     ...                       0.08902          NaN
2  84300903         M     ...                       0.08758          NaN
3  84348301         M     ...                       0.17300          NaN
4  84358402         M     ...                       0.07678          NaN

[5 rows x 33 columns]
```

By looking into the info and head of our data, here are some preliminary conclusions:

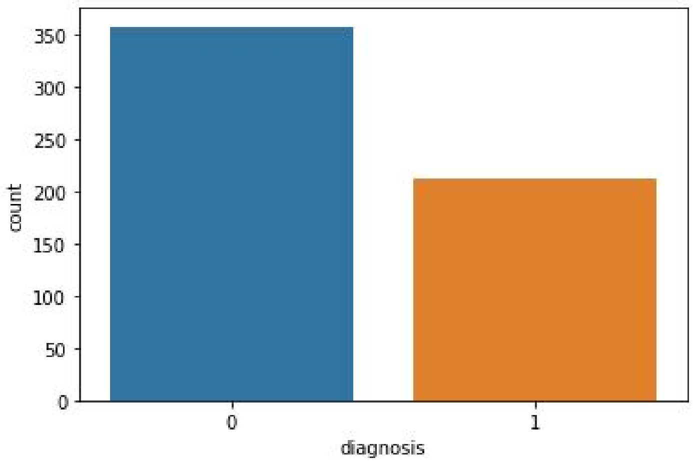
- The ‘id’ column is irrelevant to our analysis.
- The ‘Unamed:32’ column contains null value which is also irrelevant to our analysis.
- As our target value, the ‘diagnosis’ column contains value either ‘M’ or ‘B’, which shows the nature of the tumor. We need to convert these values to binary 1 and 0.

After removing these two relevant columns and converting the diagnosis column into numerical values, below is the screen show of top five rows after data cleaning.



```
In [81]: data.head()
Out[81]:
   diagnosis          ...            fractal_dimension_worst
0          1          ...                            0.11890
1          1          ...                            0.08902
2          1          ...                            0.08758
3          1          ...                            0.17300
4          1          ...                            0.07678
```

- Data Visualization

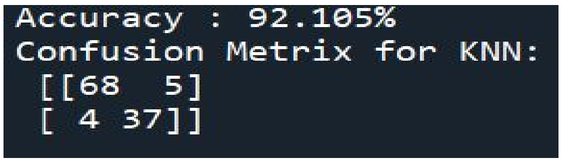


From the graph above, we can clearly see the amount of cases that are located at an early stage of breast cancer, and these cases are able to be cured completely.

- Modeling

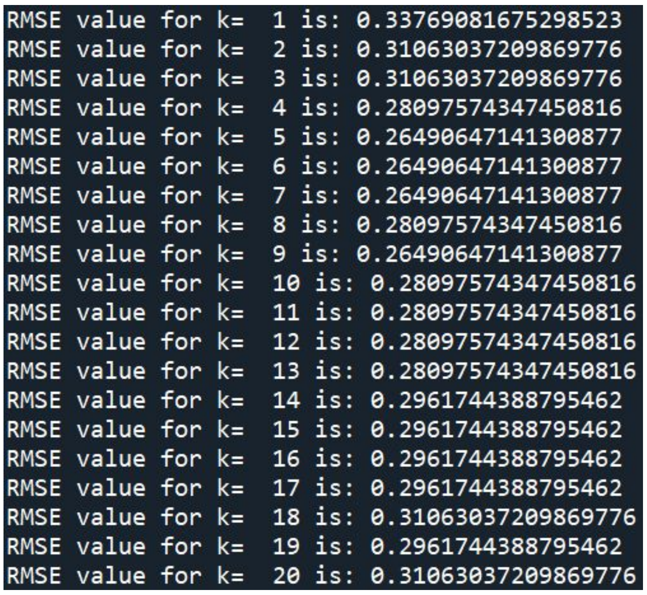
● KNN

The parameter we chose to build our KNN model is where K = 13.

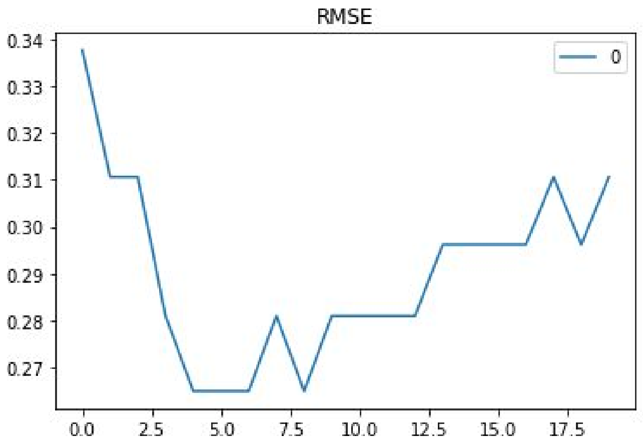


```
Accuracy : 92.105%
Confusion Metrix for KNN:
 [[68  5]
 [ 4 37]]
```

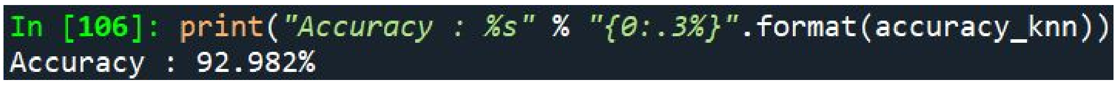
After training our model and running through evaluation, the result came with accuracy of 92.105%. However, we did not know the best K value yet, and to optimize our KNN model, we calculated for the best K based on RMSE.



```
RMSE value for k=  1 is: 0.33769081675298523
RMSE value for k=  2 is: 0.3106303720986977б
RMSE value for k=  3 is: 0.3106303720986977б
RMSE value for k=  4 is: 0.2809757434745081б
RMSE value for k=  5 is: 0.26490647141300877
RMSE value for k=  6 is: 0.26490647141300877
RMSE value for k=  7 is: 0.26490647141300877
RMSE value for k=  8 is: 0.2809757434745081б
RMSE value for k=  9 is: 0.26490647141300877
RMSE value for k=  10 is: 0.2809757434745081б
RMSE value for k=  11 is: 0.2809757434745081б
RMSE value for k=  12 is: 0.2809757434745081б
RMSE value for k=  13 is: 0.2809757434745081б
RMSE value for k=  14 is: 0.2961744388795462
RMSE value for k=  15 is: 0.2961744388795462
RMSE value for k=  16 is: 0.2961744388795462
RMSE value for k=  17 is: 0.2961744388795462
RMSE value for k=  18 is: 0.3106303720986977б
RMSE value for k=  19 is: 0.2961744388795462
RMSE value for k=  20 is: 0.3106303720986977б
```



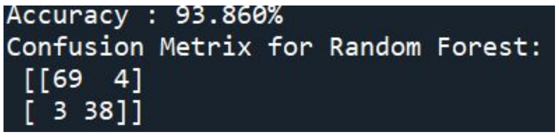
After calculating the RMSE based on different K values, we chose **5** as our optimized K value. In this case, our accuracy increased from **92.105%** to **92.982%**



```
In [106]: print("Accuracy : %s" % "{0:.3%}".format(accuracy_knn))
Accuracy : 92.982%
```

- **Random Forest**

After the KNN model, our next step moved to the random forest model. Same as training the KNN model, we need to find out the highest accuracy in breast cancer prediction. Similarly, we still randomly split the dataset as 80% for training and 20% for testing. The RF model yields an accuracy of **93.860%** without further optimizations for the random forest model.

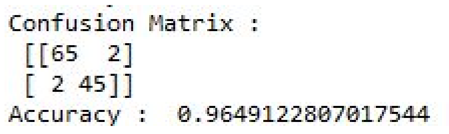


```
Accuracy : 93.860%
Confusion Metrix for Random Forest:
 [[69  4]
 [ 3 38]]
```

- ## Logistic Regression

In this project, we also used a logistic regression model to make the prediction, and since the logistic regression model is able to show the relationship between example values and indicator variables, we think it could be helpful for our analysis.

Differ from the previous model, we split the dataset as 67% as training dataset and 33% as testing dataset randomly. After we split the data, we applied the standard scaler for both training and testing dataset to make sure the accuracy of our result.



```
Confusion Matrix :
 [[65  2]
 [ 2 45]]
Accuracy :  0.9649122807017544
```

From the logistic regression model, the accuracy was **97.872%** which is really high and looks good.

- ## ANN

For breast cancer prediction, we have tried Artificial neural networks. ANN has been widely used on medical diagnosis, so we hope this model will have a good performance on the breast cancer prediction.

The ANN model's performance is highly dependent on the number of hidden layers and the nodes on each layer. So, to create a reliable ANN model, we have to find the most efficient combination of the hidden layer numbers and node numbers.

Model Construction

Activation

Besides the hidden layer numbers and nodes numbers, we have decided to use 'ReLU' as the activation.

First of all, we excluded 'softmax' activation from our choices. According to the dataset and the goal of our model, the problem is more like a classification problem, so that we do not expect that 'softmax' activation will have a good performance in our model.

Second, compared to 'sigmoid' activation, ReLU works better with more layers because it overcomes the vanishing gradient problem. Furthermore, because ReLU only needs to pick the max values during the process, it will make the model run faster than the 'sigmoid' activation. The benefit for us is that the faster the model learns, the more combinations we will be able to test.

Optimizer

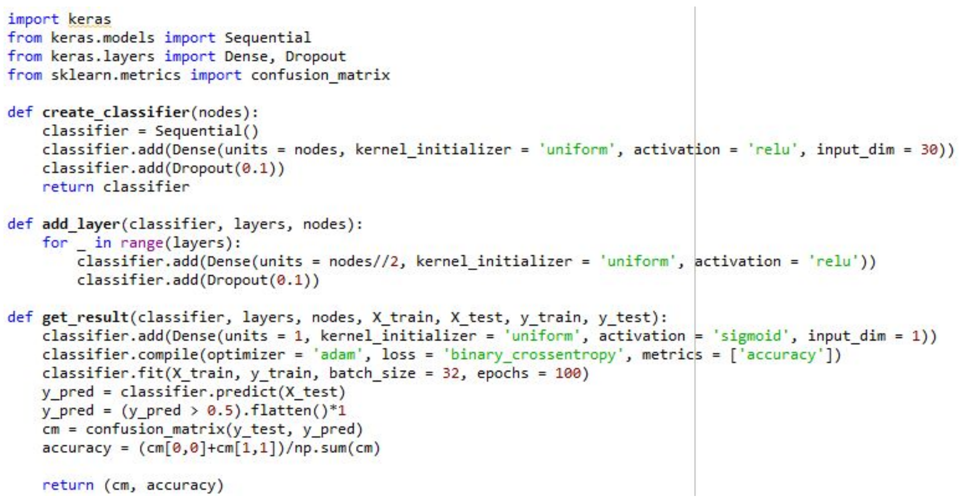
For the optimizer, we simply chose ‘adam’ as the optimizer, because it is widely used in neural networks and it included some very good properties from other optimization algorithms. It is reliable.

Furthermore, ‘adam’ also requires less memory than most other optimization algorithms and relatively runs faster.

Batch Size

We choose 32 as the batch size because the dataset is not very big, the dataset only contains 569 rows (455 training rows after splitting).

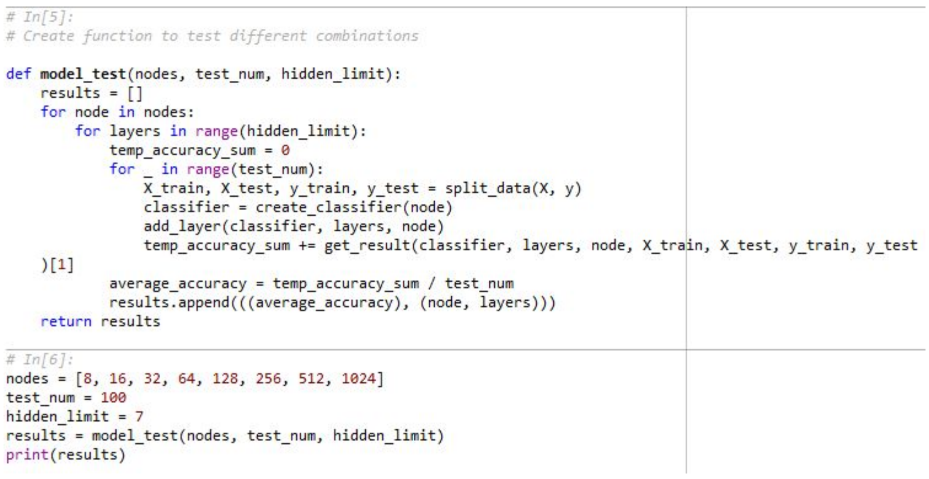
Final Model



```python
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import confusion_matrix

def create_classifier(nodes):
    classifier = Sequential()
    classifier.add(Dense(units = nodes, kernel_initializer = 'uniform', activation = 'relu', input_dim = 30))
    classifier.add(Dropout(0.1))
    return classifier

def add_layer(classifier, layers, nodes):
    for _ in range(layers):
        classifier.add(Dense(units = nodes//2, kernel_initializer = 'uniform', activation = 'relu'))
        classifier.add(Dropout(0.1))

def get_result(classifier, layers, nodes, X_train, X_test, y_train, y_test):
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid', input_dim = 1))
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
    classifier.fit(X_train, y_train, batch_size = 32, epochs = 100)
    y_pred = classifier.predict(X_test)
    y_pred = (y_pred > 0.5).flatten()*1
    cm = confusion_matrix(y_test, y_pred)
    accuracy = (cm[0,0]+cm[1,1])/np.sum(cm)

    return (cm, accuracy)
```

Testing Description

To find out the best combination, we are going to test from the set that the initial layer has 8, 16, 32, 64, 128, 265, 512, 1024 nodes, and the hidden layers have a half number of initial layers nodes. We will also try the impact of different hidden layers (from 0 hidden layer to 5 hidden layers).



```
# In[5]:
# Create function to test different combinations

def model_test(nodes, test_num, hidden_limit):
    results = []
    for node in nodes:
        for layers in range(hidden_limit):
            temp_accuracy_sum = 0
            for _ in range(test_num):
                X_train, X_test, y_train, y_test = split_data(X, y)
                classifier = create_classifier(node)
                add_layer(classifier, layers, node)
                temp_accuracy_sum += get_result(classifier, layers, node, X_train, X_test, y_train, y_test
    )[1]
            average_accuracy = temp_accuracy_sum / test_num
            results.append(((average_accuracy), (node, layers)))
    return results

# In[6]:
nodes = [8, 16, 32, 64, 128, 256, 512, 1024]
test_num = 100
hidden_limit = 7
results = model_test(nodes, test_num, hidden_limit)
print(results)
```

Sample Test Results

First Test (ranking by accuracy):

ANN model with 2 hidden layers. The first layer has 512 nodes and all hidden layers have 256 nodes. Accuracy: 0.9824561403508771

ANN model with 5 hidden layers. The first layer has 8 nodes and all hidden layers have 4 nodes.

Accuracy: 0.9736842105263158

ANN model with 4 hidden layers. The first layer has 16 nodes and all hidden layers have 8 nodes. Accuracy: 0.9736842105263158

Second Test (ranking by accuracy):

ANN model with 2 hidden layers. The first layer has 128 nodes and all hidden layers have 64 nodes. Accuracy: 0.9912280701754386

ANN model with 2 hidden layers. The first layer has 256 nodes and all hidden layers have 128 nodes. Accuracy: 0.9736842105263158

ANN model with 4 hidden layers. The first layer has 1024 nodes and all hidden layers have 512 nodes. Accuracy: 0.9736842105263158

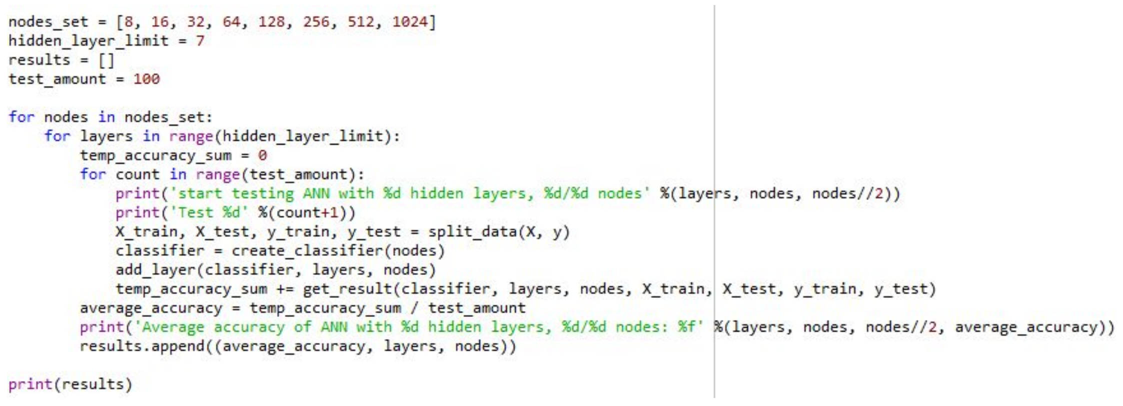
Third Test (ranking by accuracy):

ANN model with 3 hidden layers. The first layer has 1024 nodes and all hidden layers have 512 nodes. Accuracy: 0.9824561403508771

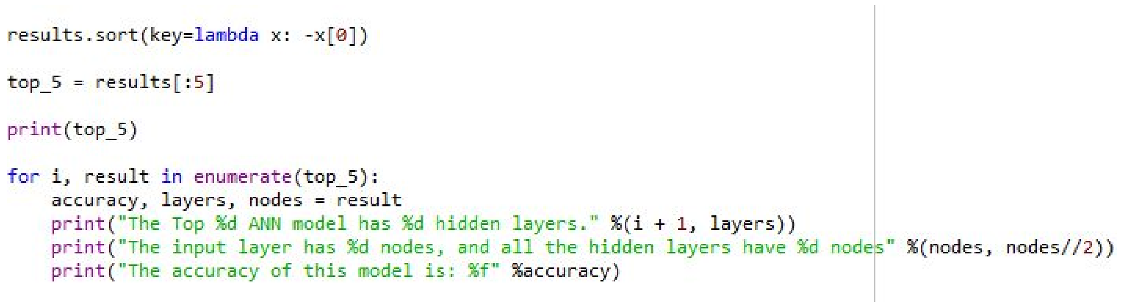
ANN model with 1 hidden layer. The first layer has 64 nodes and all hidden layers have 32 nodes. Accuracy: 0.9736842105263158

ANN model with 5 hidden layers. The first layer has 128 nodes and all hidden layers have 64 nodes. Accuracy: 0.9736842105263158

Final Test Result



```python
nodes_set = [8, 16, 32, 64, 128, 256, 512, 1024]
hidden_layer_limit = 7
results = []
test_amount = 100

for nodes in nodes_set:
    for layers in range(hidden_layer_limit):
        temp_accuracy_sum = 0
        for count in range(test_amount):
            print('start testing ANN with %d hidden layers, %d/%d nodes' %(layers, nodes, nodes//2))
            print('Test %d' %(count+1))
            X_train, X_test, y_train, y_test = split_data(X, y)
            classifier = create_classifier(nodes)
            add_layer(classifier, layers, nodes)
            temp_accuracy_sum += get_result(classifier, layers, nodes, X_train, X_test, y_train, y_test)
        average_accuracy = temp_accuracy_sum / test_amount
        print('Average accuracy of ANN with %d hidden layers, %d/%d nodes: %f' %(layers, nodes, nodes//2, average_accuracy))
        results.append((average_accuracy, layers, nodes))

print(results)
```



```python
results.sort(key=lambda x: -x[0])

top_5 = results[:5]

print(top_5)

for i, result in enumerate(top_5):
    accuracy, layers, nodes = result
    print("The Top %d ANN model has %d hidden layers." %(i + 1, layers))
    print("The input layer has %d nodes, and all the hidden layers have %d nodes" %(nodes, nodes//2))
    print("The accuracy of this model is: %f" %accuracy)
```

So far, we have done the test 100 times for each combination, and we've got the 5 **most accurate combination** among all of the choices, they are:

- The Top 1 ANN model has 0 hidden layers.
    - The input layer has 128 nodes, and all the hidden layers have 64 nodes
    - The accuracy of this model is: **0.967544**
- The Top 2 ANN model has 1 hidden layer.
    - The input layer has 64 nodes, and all the hidden layers have 32 nodes
    - The accuracy of this model is: **0.963158**
- The Top 3 ANN model has 0 hidden layers.
    - The input layer has 256 nodes, and all the hidden layers have 128 nodes
    - The accuracy of this model is: **0.963158**
- The Top 4 ANN model has 2 hidden layers.
    - The input layer has 512 nodes, and all the hidden layers have 256 nodes
    - The accuracy of this model is: **0.963158**
- The Top 5 ANN model has 2 hidden layers.
    - The input layer has 1024 nodes, and all the hidden layers have 512 nodes
    - The accuracy of this model is: **0.963158**

The 5 **most inaccurate combinations** are:

- The Worst 1 ANN model has 3 hidden layers.

- The input layer has 8 nodes, and all the hidden layers have 4 nodes
- The accuracy of this model is: **0.879825**

- The Worst 2 ANN model has 3 hidden layers.
  - The input layer has 16 nodes, and all the hidden layers have 8 nodes
  - The accuracy of this model is: **0.878070**
- The Worst 3 ANN model has 5 hidden layers.
  - The input layer has 8 nodes, and all the hidden layers have 4 nodes
  - The accuracy of this model is: **0.846491**
- The Worst 4 ANN model has 4 hidden layers.
  - The input layer has 8 nodes, and all the hidden layers have 4 nodes
  - The accuracy of this model is: **0.810526**
- The Worst 5 ANN model has 6 hidden layers.
  - The input layer has 8 nodes, and all the hidden layers have 4 nodes
  - The accuracy of this model is: **0.772807**

Conclusion

Based on comparison and our observation, the ANN model with 0 hidden layers and 128 nodes in the input layer has the best accuracy so far, which is **96.75%.** And the second best one is logistic regression, which has **96.49%** accuracy.

Even though the accuracy of the best ANN model is **0.36%** higher than logistic regression, logistic regression has a big advantage that it runs much faster than the ANN model. To complete the ANN model on our local machine we spent about 3 days to complete the learning process. But for logistic regression, even if we have tried to test it hundreds of times, the results will be returned within a few seconds, meaning that logistic regression will be able to be fastly adapted and implemented.

However, as a medical problem prediction, accuracy would be the most important part that we have to consider. However, if the dataset becomes huge or even overwhelming, the learning process of neural networks would take so much time and resources.

So, the recommendation from our research is that we still aim to the ANN model against the problem, but before the learning process successfully completed and fully tested, we can use logistic regression to help the medical staffs to improve the breast cancer prediction process and use the model to give out valuable advice and suggestions.