Programming 3 Report

Problem 1: Explain it!

In this problem, I used diabete set from UCI Repository of machine learning databases (Newman et al. 1998).In this dataset, there are 8 Independent variables: pregnant, glucose, pressure, triceps, insulin, mass, pedigree, age, and 1 dependent variable: diabetes(positive/negative). As for the deep learning model, I implemented MLP classifier of sklearn. And as for the explainable model, I used LIME model.

1.1 Binarizing the dependent variable

In order to proceed to model fitting, digitizing the dependent variable is necessary. After that, the dependent variables became 0 for negative and 1 for positive.

```
diabetes["diabetes"] = diabetes["diabetes"].map({"neg":0, "pos":1})
diabetes["diabetes"].value_counts()
```

```
0    262
1    130
Name: diabetes, dtype: int64
```

1.2 Standardization and splitting the dataset

In this part, I standardized the data and divide the dataset into two part: train and test.

```
y = diabetes['diabetes']
X = diabetes[['pregnant', 'glucose', 'pressure', 'triceps', 'insulin', 'mass', 'pedigree', 'age']]
X_featurenames = X.columns

X_mean = X.mean(axis=0)
X_std = X.std(axis=0)
X_scaled = (X-X_mean)/X_std

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2)
```

1.3 MLPClassifier

I used function MLPClassifier from sklearn, and set the maximum iteration 500 times. The accuracy of test set is 0.7722.

```
mlp=MLPClassifier(solver="adam",max_iter=500).fit(X_train,y_train)
print("Accuracy of training: "+str(accuracy_score(y_train,mlp.predict(X_train))))
print("Accuracy of testing: "+str(accuracy_score(y_test,mlp.predict(X_test))))
```

```
Accuracy of training: 0.9137380191693291
Accuracy of testing: 0.7721518987341772
```

The number of layer is 3.

```
print('The number of layers is :', mlp.n_layers_)
```

```
The number of layers is : 3
```

Here is the classification report.

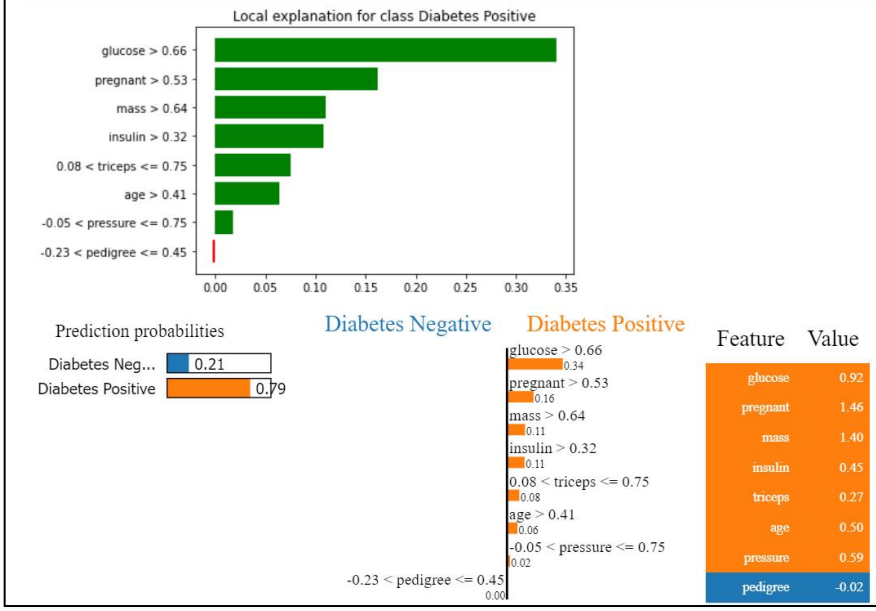
```
model = mlp
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, target_names=["Diabetes Negative", "Diabetes Positive"]))
```

```
                  precision    recall  f1-score   support

Diabetes Negative       0.85      0.82      0.83        55
Diabetes Positive       0.62      0.67      0.64        24

        accuracy                           0.77        79
       macro avg       0.73      0.74      0.74        79
    weighted avg       0.78      0.77      0.77        79
```

1.4 Five examples of positive

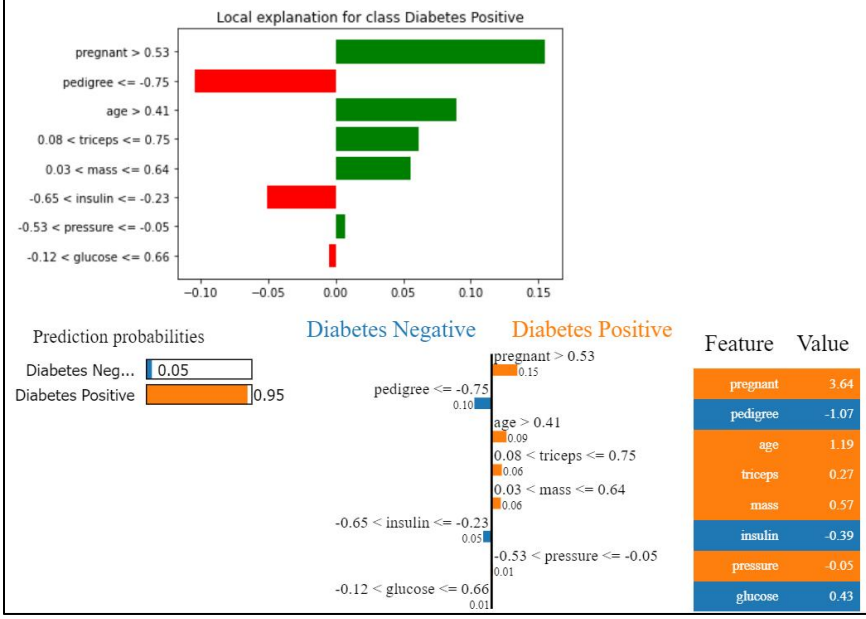
```
Example 1
Intercept 0.15323317014456783
Prediction_local [1.02826825]
Right: 0.7923309496097739
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPCla
fier was fitted with feature names
  warnings.warn(
```



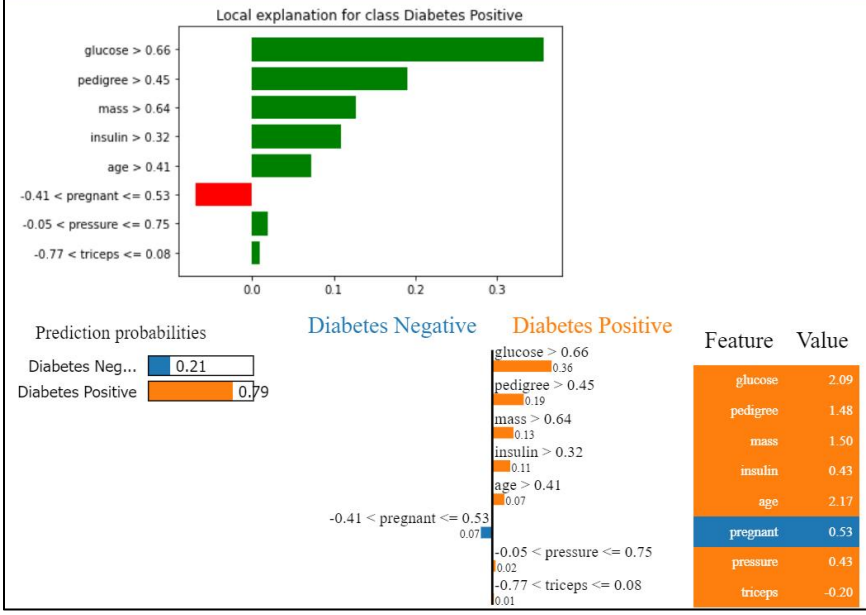

```
Example 2
Intercept 0.3170452654133104
Prediction_local [0.524604]
Right: 0.9485947350909206
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



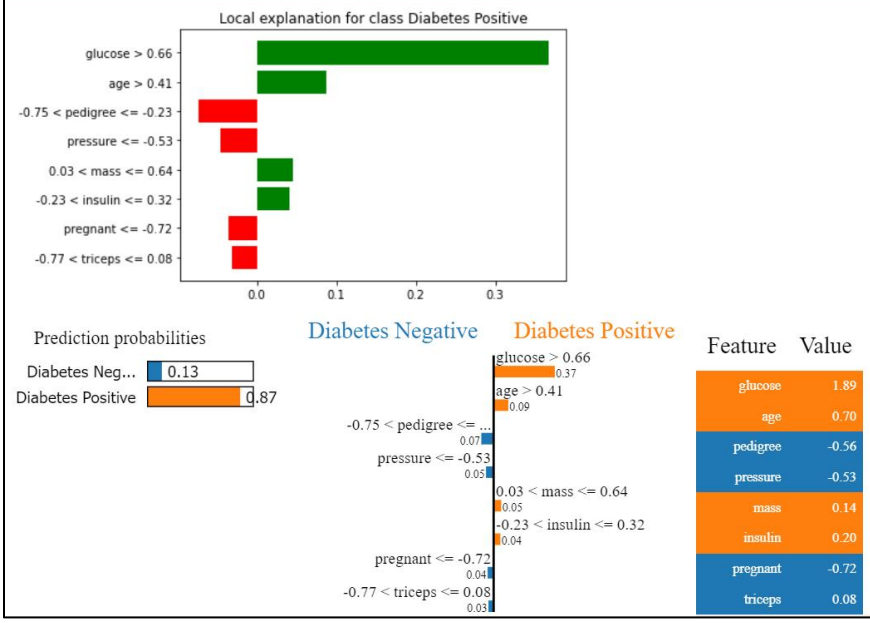
```
Example 3
Intercept 0.15205443633882787
Prediction_local [0.96941777]
Right: 0.7946081258551899
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



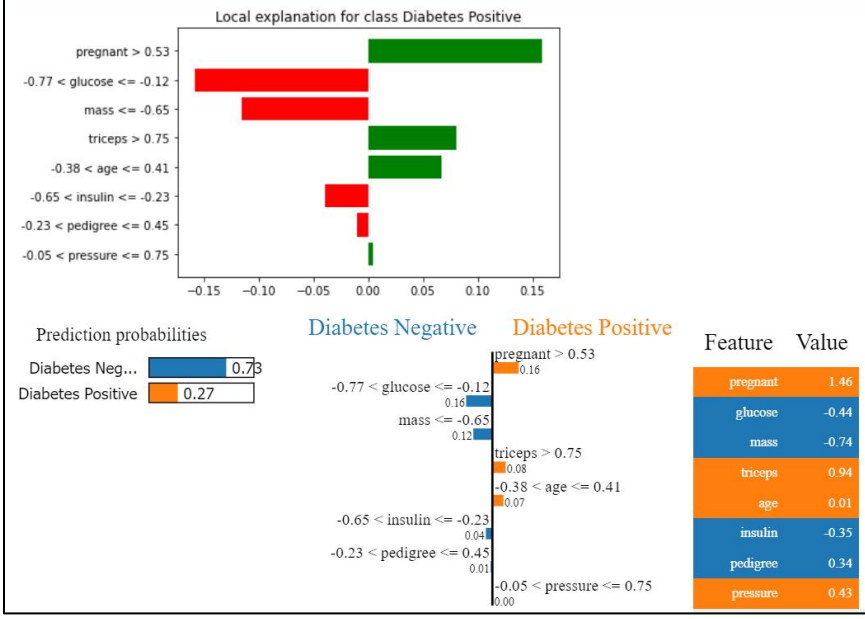

```
Example 4
Intercept 0.28975252895948994
Prediction_local [0.64402204]
Right: 0.8664928474401244
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPCla
fier was fitted with feature names
  warnings.warn(
```



```
Example 5
Intercept 0.376578248235281
Prediction_local [0.36328658]
Right: 0.27134791866716695
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



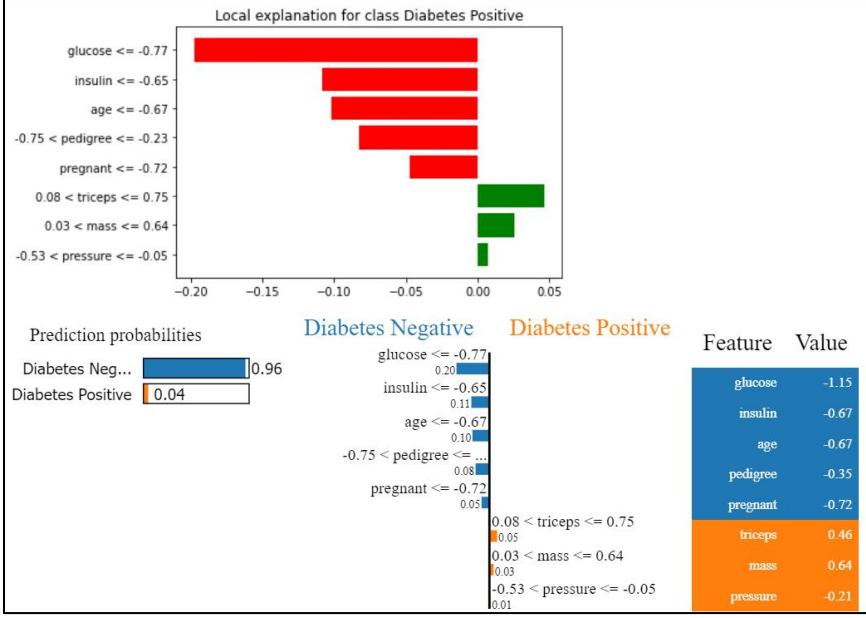
These five examples are positive explanations for the outcome of the MLPClassifer. Let the first one as an instance, the output shows that the LIME model has an intercept of 0.1532 and the LIME model predicted a value of 1.0283 (Prediction_local). The MLPclassifier model predicted positive probability of 0.7923. In this figure, the green bar on the right represents support for diabetes positive, while the red bar on the left represents support for diabetes negative. For selected examples of observations, 'glucose' > 0.66 (data after normalization) showed the highest support for diabetes positivity. In other words, for this example where 'glucose' > 0.66, it mostly leads to diabetes positive.

But I have to say that this model does not fit well. For example, in the last example, the model actually thinks that this example has an 73% probability of being negative, but it is completely positive.

1.5 Five examples of negative

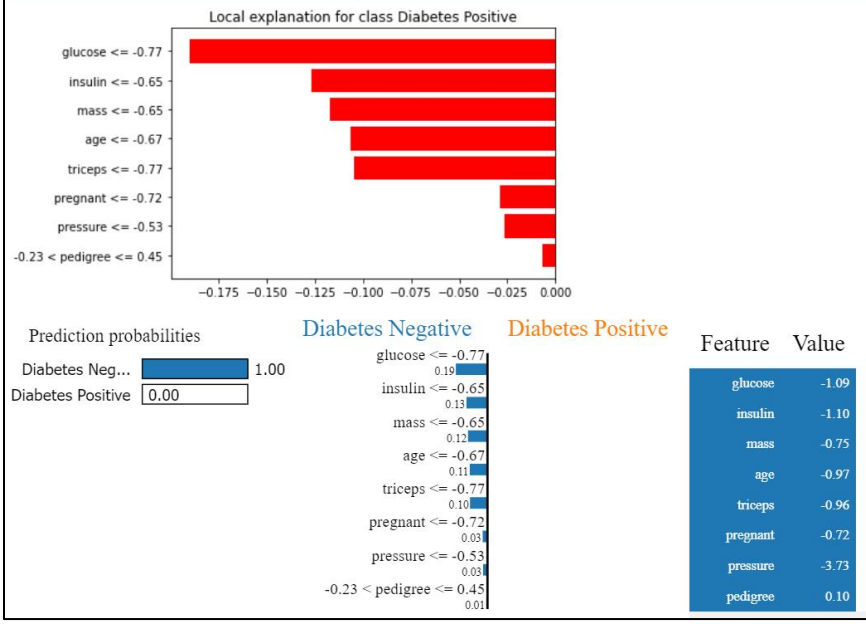
```
Example 1
Intercept 0.4965007449607762
Prediction_local [0.03710863]
Right: 0.04483518160669648
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



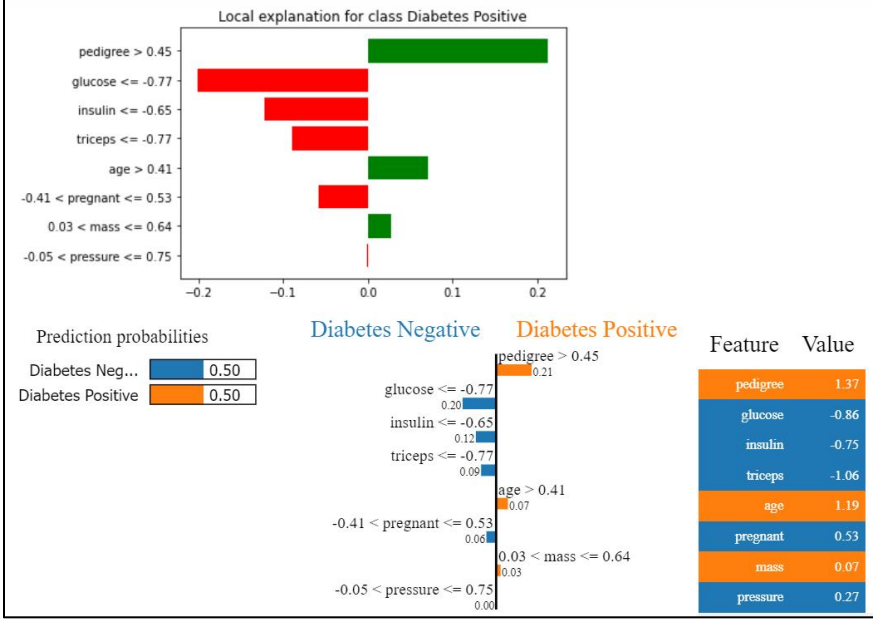

```
Example 2
Intercept 0.5504329113507633
Prediction_local [-0.15796325]
Right: 0.003089295203262389
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



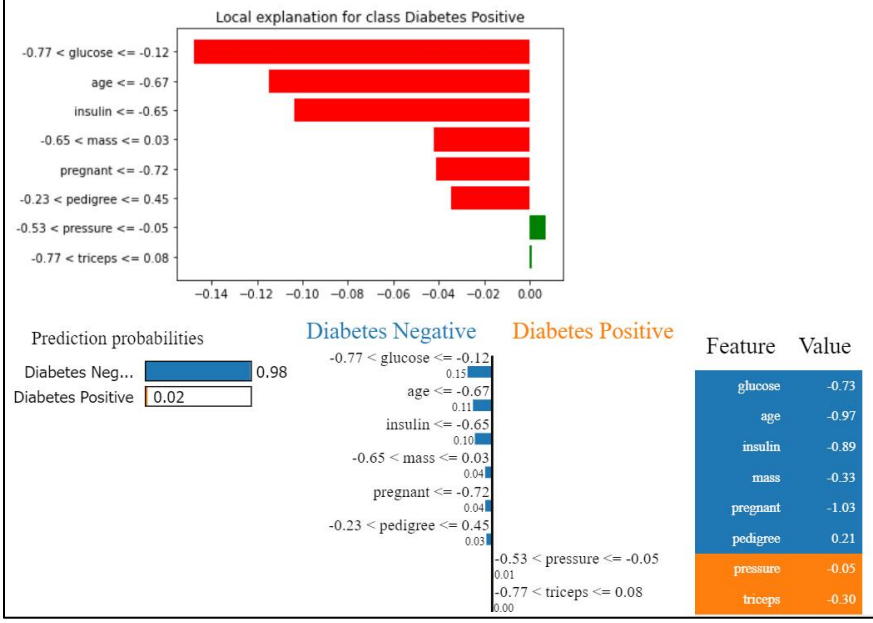
```
Example 3
Intercept 0.40655561545145436
Prediction_local [0.2439812]
Right: 0.49991832418856424
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPCla
fier was fitted with feature names
  warnings.warn(
```



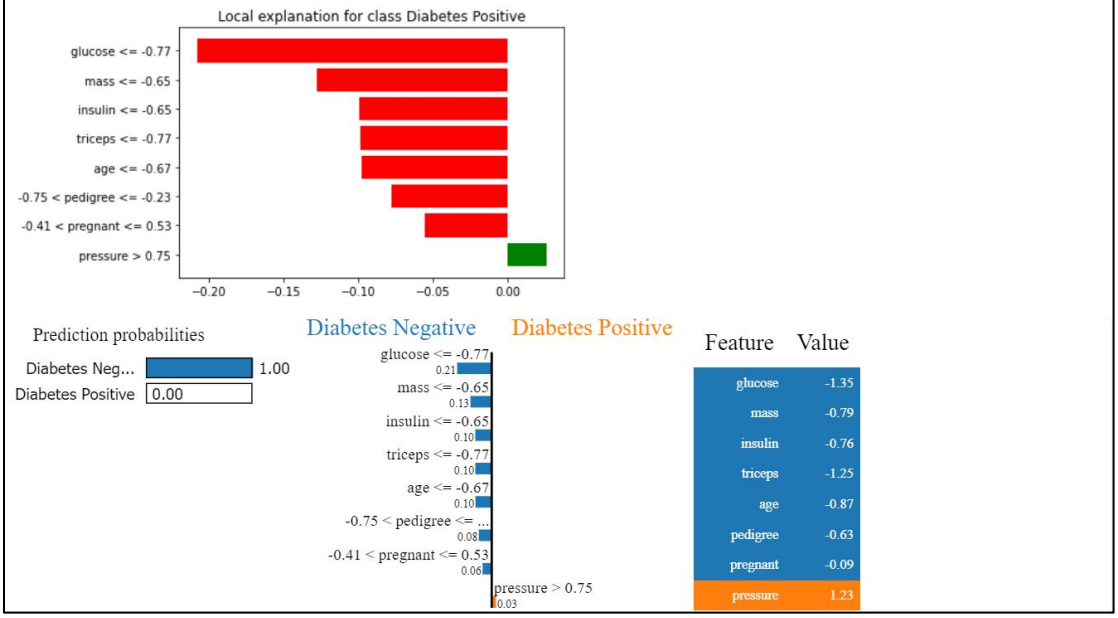

```
---------------------------
Example 4
Intercept 0.4941987869301442
Prediction_local [0.0183463]
Right: 0.019938853950555273
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPCla
fier was fitted with feature names
  warnings.warn(
```



```
Example 5
Intercept 0.5597894971380504
Prediction_local [-0.17757887]
Right: 8.955036238262458e-05
```

```
C:\Users\gxysi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but MLPClas
fier was fitted with feature names
  warnings.warn(
```



Similarly, these five examples are negative explanations. For the first one, the LIME model has an intercept of 0.4965 and the LIME model predicted a value of 0.0371 (Prediction_local). The MLPclassifier model predicted the positive probability of 0.0448. In this figure, the green bar on the right represents support for diabetes positive, while the red bar on the left represents support for diabetes negative. For selected examples of observations, 'glucose' <= -0.77 (data after normalization) showed the highest support for diabetes negativity. In other words, for this example where 'glucose' <= -0.77, it mostly leads to diabetes negative.

1.6 Predictions

```
1  mlp_pred_1 = mlp.predict(X_example_1)
2  mlp_pred_0 = mlp.predict(X_example_0)
3  print('The predictions of 5 positive examples:',mlp_pred_1)
4  print('The predictions of 5 negative examples:',mlp_pred_0)
```

```
The predictions of 5 positive examples: [1 1 1 1 0]
The predictions of 5 negative examples: [0 0 0 0 0]
```

For the negatives instances, all of the examples are correct, while the positives examples have one incorrect prediction ,unfortunately.

Problem 2: Random Forest

In this section, I used RandomForestClassifier, and GridSearchCV from sklearn.

2.1 GridSearchCV

In order to get the best parameters, I implemented grid search cross validation.

```
model = RandomForestClassifier()
param_grid = {
    'n_estimators': [200,500,1000],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : range(20),
    'criterion' :['gini', 'entropy']
}
cv = GridSearchCV(model,param_grid,cv=5)
cv.fit(X_train, y_train)
```

And I got the best parameters after fitting.

```
cv.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'auto',
 'n_estimators': 1000}
```

2.2 RandomForestClassifier

I used parameters obtained by grid searching, fitted the model and got the report. For testing, the accuracy is 0.83545, which is higherr than mlp.

```
model = RandomForestClassifier(criterion='gini',
                               max_depth=5,
                               max_features='auto',
                               n_estimators = 1000)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("Accuracy of training: "+str(accuracy_score(y_train,model.predict(X_train))))
print("Accuracy of testing: "+str(accuracy_score(y_test,model.predict(X_test))))
print(classification_report(y_test, y_pred, target_names=["Diabetes Negative", "Diabetes Positive"]))
```

```
Accuracy of training: 0.9201277955271565
Accuracy of testing: 0.8354430379746836
                  precision    recall  f1-score   support

Diabetes Negative       0.88      0.89      0.88        55
Diabetes Positive       0.74      0.71      0.72        24

        accuracy                           0.84        79
       macro avg       0.81      0.80      0.80        79
    weighted avg       0.83      0.84      0.83        79
```

2.3 The best tree

I tried to find the best tree in forest and made predictions.

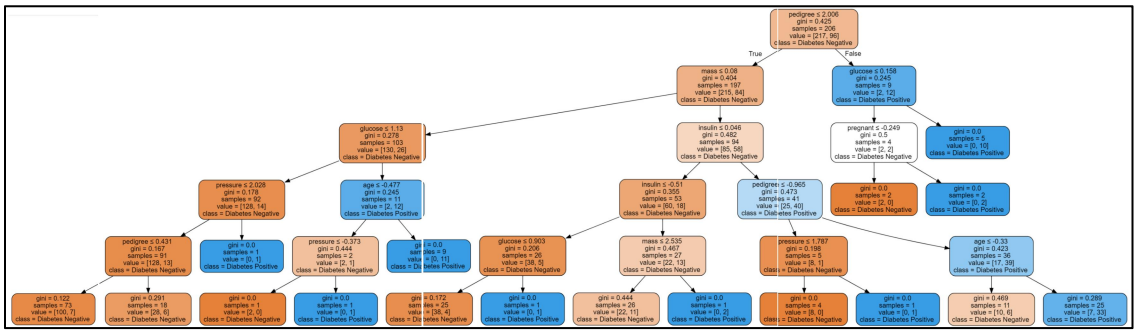
```
estimator_accuracy = []
n_estimators = 1000
for cur_estimator in range(n_estimators):
    estimator_accuracy.append([cur_estimator,
                               accuracy_score(y_test, model.estimators_[cur_estimator].predict(X_test))])

estimator_accuracy=pd.DataFrame(estimator_accuracy,
                                columns=['estimator_number','accuracy'])

estimator_accuracy.sort_values(inplace=True,
                               by='accuracy',
                               ascending=False)

best_tree= model.estimators_[estimator_accuracy.head(1)['estimator_number'].values[0]]
```

```
best_tree_pred_1 = best_tree.predict(X_example_1)
best_tree_pred_0 = best_tree.predict(X_example_0)
print('The predictions of 5 positive examples:',best_tree_pred_1)
print('The predictions of 5 negative examples:',best_tree_pred_0)
```

```
The predictions of 5 positive examples: [1. 1. 1. 1. 0.]
The predictions of 5 negative examples: [0. 0. 0. 0. 0.]
```

For the best decision tree in the random forest, I got the same result as MLP, which means the predictions of negativity are correct ,but the last one in positivity is wrong. It's kind of tricky, it seems to mislead the model to make a wrong judgment.

And the graph of the best tree:

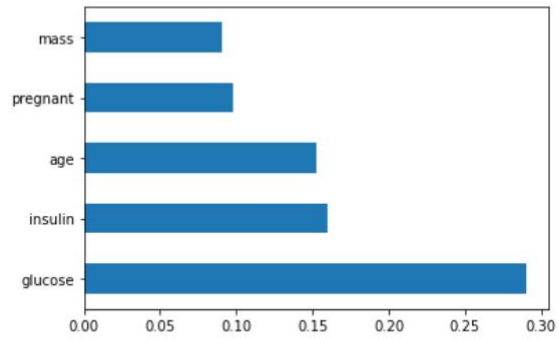


Problem 3: Feature correlations

Top 5 features are mass, pregnant, age, insulin, and glucose.I suppose this explanation is more readable and clear. We can visually see the importance comparison between features, and can also rank the importance of features. Such an intuitive concept map is very practical.

```
feat_importances = pd.Series(model.feature_importances_, index = X_featurenames)
feat_importances.nlargest(5).plot(kind = 'barh')
```

```
<AxesSubplot:>
```



Top 5 features are mass, pregnant, age, insulin, and glucose.

I suppose this explanation is more readable and clear.

We can visually see the importance comparison between features, and can also rank the importance of features.

Such an intuitive concept map is very practical.