EMORY UNIVERSITY

Introduction to Business Analytics [MSBA]

Homework #2

_____Jie Zhu_____

(put your full name above (incl. any nicknames))

Note: This is an individual homework. Discussing this homework with your classmates is a violation of the Honor Code. If you borrow code from somewhere else, please add a comment in your code to make it clear what the source of the code is (e.g., a URL would sufficient). If you borrow code and you don't provide the source, it is a violation of the Honor Code.

Total grade: ________ out of ___150___ points

1) (15 points) Would you frame the problem of e-mail spam detection as a supervised learning problem or an unsupervised learning problem? Please justify your answer.

I would frame the problem of e-mail spam detection as a supervised learning problem. This is because this problem has an obvious binary target – to determine whether the mail is spam or not. Both spam samples and non-spam samples should be used to train the model. This problem has a specific purpose and requires data on the target to solve. Thus, it's a classic supervised learning problem.

2) (15 points) What is a test set and why would you want to use it?

A test set is generated by splitting the original dataset into training set and test set. It is used to evaluate the accuracy of the model trained by the training set. We could only get in-sample accuracy from training set alone. It's possible that the model has over-fitting problems, and these problems can only be detected by using test set. In short, we use test set to find the "sweet spot" where the performance of test set result is maximized.

3) (20 points) What are the similarities and differences of decision trees and logistic regression? When might you prefer to use one over another?

Decision trees and logistic regression both use decision boundaries to classify. However, they are different in some ways. Decision trees use decision boundaries that are perpendicular to the instance-space axes, while logistic regression can use decision boundaries of any direction or orientation. Moreover, decision trees select a single attribute at a time, while linear classifiers use a weighted combination instead. Finally, decision trees can cut up the instance space arbitrarily finely into very small regions, while logistic regression can only place a single decision surface.

I prefer to use decision trees over logistic regression when I have large training-set, however, I prefer to use logistic regression over decision trees when the training-set is small. This is because logistic regression yields better generalization accuracy than decision trees for small training-set. For larger training set, decision trees are more flexible than logistic regression and can represent nonlinear relationships.

4) (30 points) You have a fraud detection task (predicting whether a given credit card transaction is "fraud" vs. "non-fraud") and you built a classification model for this purpose. For any credit card transaction, your model estimates the probability that this transaction is "fraud". The following table represents the probabilities that your model estimated for the validation dataset containing 10 records.

Actual Class (from validation data)	Estimated Probability of Record Belonging to Class "fraud"
fraud	0.95
fraud	0.91
fraud	0.75
non-fraud	0.67
fraud	0.61
non-fraud	0.46
fraud	0.42
non-fraud	0.25
non-fraud	0.09
non-fraud	0.04

Based on the above information, answer the following questions:

a) **What is the overall accuracy of your model, if the chosen probability cutoff value is 0.3? What is the overall accuracy of your model, if the chosen probability cutoff value is 0.8?**

Actual Class (from validation data)	Prediction (cutoff = 0.3)
fraud	fraud
fraud	fraud
fraud	fraud
non-fraud	fraud
fraud	fraud
non-fraud	fraud
fraud	fraud
non-fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud

Accuracy(cutoff=0.3) = 1 – 2/10 = 0.8

Actual Class (from validation data)	Prediction (cutoff = 0.8)
fraud	fraud
fraud	fraud
fraud	non-fraud
non-fraud	non-fraud
fraud	non-fraud
non-fraud	non-fraud
fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud

Accuracy(cutoff=0.8) = 1 – 3/10 = 0.7

b) **What probability cutoff value should you choose, in order to have Precision fraud = 100% for your model? (Explain.) What is the overall accuracy of your model in this case?**

Cutoff value should be greater than 0.67. Precision fraud = TP/(TP+FP) = 3/(3+0) = 100%.

Actual Class (from validation data)	**Prediction (cutoff = 0.7)**
fraud	fraud
fraud	fraud
fraud	fraud
non-fraud	non-fraud
fraud	non-fraud
non-fraud	non-fraud
fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud

Accuracy(cutoff=0.7) = 1 – 2/10 = 0.8

c) **What probability cutoff value should you choose, in order to have Recall fraud = 100% for your model? (Explain.) What is the overall accuracy of your model in this case?**

Cutoff value should be less than 0.42. Recall fraud = TP/(TP+FN) = 5/(5+0) = 100%.

Actual Class (from validation data)	**Prediction (cutoff = 0.4)**
fraud	fraud
fraud	fraud
fraud	fraud
non-fraud	fraud
fraud	fraud
non-fraud	fraud
fraud	fraud
non-fraud	non-fraud
non-fraud	non-fraud
non-fraud	non-fraud

Accuracy(cutoff=0.4) = 1 – 2/10 = 0.8

5) (70 points) [Mining publicly available data. Please implement the following models both with Rapidminer and Python but explore the data (e.g., descriptive statistics etc.) just with Python]

Please use the dataset on breast cancer research from this link: http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data [Note: Rapidminer can import .data files in the same way it can import .csv files. For Python please read the data directly from the URL without downloading the file on your local disk.**] The description of the data and attributes can be found at this link: http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names. Each record of the data set represents a different case of breast cancer. Each case is described with 30 real-valued attributes: attribute 1 represents case id, attributes 3-32 represent various physiological characteristics, and attribute 2 represents the type (benign or malignant). If the dataset has records with missing values, you can filter out these records using Python and/or Rapidminer before proceeding with the models. Alternatively, if the data set has missing values, you could infer the missing values.**

Perform a predictive modeling analysis on this same dataset using the a) k-NN technique (for k=3) and b) Logistic Regression. Please be specific about what other parameters you specified for your models.

Present a brief overview of your predictive modeling process, explorations, and discuss your results.

Compare the k-NN model with the Logistic Regression model: which performs better? Make sure you present information about the model "goodness" (i.e., confusion matrix, predictive accuracy, precision, recall, f-measure). Please be clear about any assumptions you might make when you choose the best performing model.

Please show screenshots of the models you have built with Rapidminer and Python, show screenshots of the performance results, and the parameters you have specified.

I. Descriptive Statistics

After importing the dataset and renaming the column names, I find that there's no null value in this dataset.

```
data.isnull().sum().sum()
```

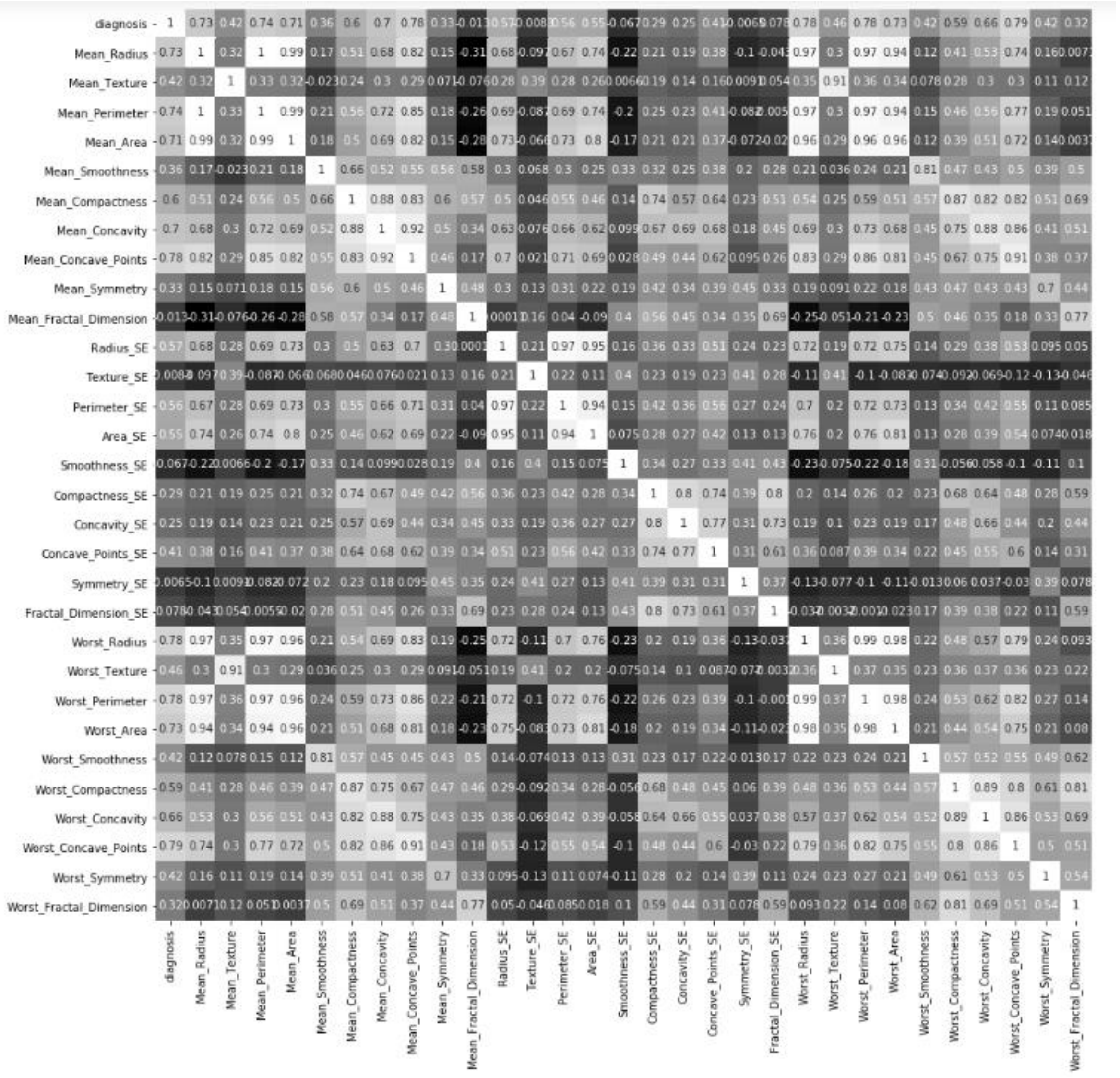
```
0
```

Following are some basic statistics. There are 569 records in total. Among all 32 columns, the second column "diagnosis" is our target variable, which is binary, 1 for malignant and 0 for benign.

	ID	diagnosis	Mean_Radius	Mean_Texture	Mean_Perimeter	Mean_Area	Me
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	0.000000	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	0.000000	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	0.000000	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	1.000000	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	1.000000	28.110000	39.280000	188.500000	2501.000000	

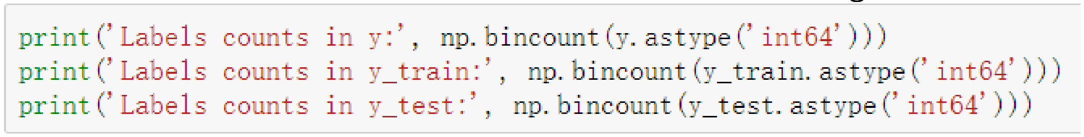
8 rows × 32 columns

The heat map of the dataset shows that there are some highly correlated variables. We will discuss the effect of multicollinearity after we have the result of the initial models.

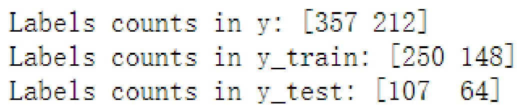


II. Split Dataset

For both KNN model and LR model, I use 70% data as training set and 30% data as test set.



```
print('Labels counts in y:', np.bincount(y.astype('int64')))
print('Labels counts in y_train:', np.bincount(y_train.astype('int64')))
print('Labels counts in y_test:', np.bincount(y_test.astype('int64')))
```

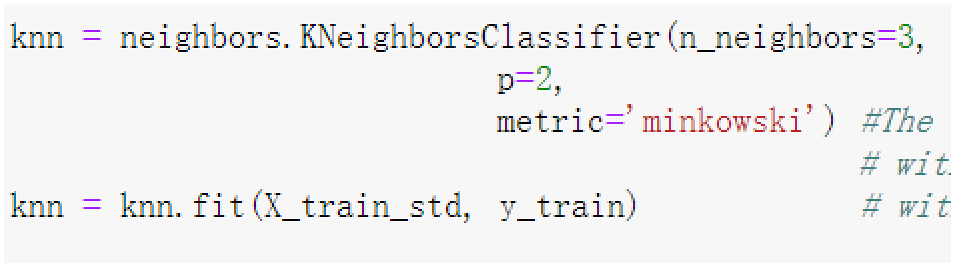


```
Labels counts in y: [357 212]
Labels counts in y_train: [250 148]
Labels counts in y_test: [107  64]
```

Among all 569 data points, 357 are malignant and 212 are benign. In the training set, 250 are malignant and 148 are benign. In the test set 107 are malignant and 64 are benign.

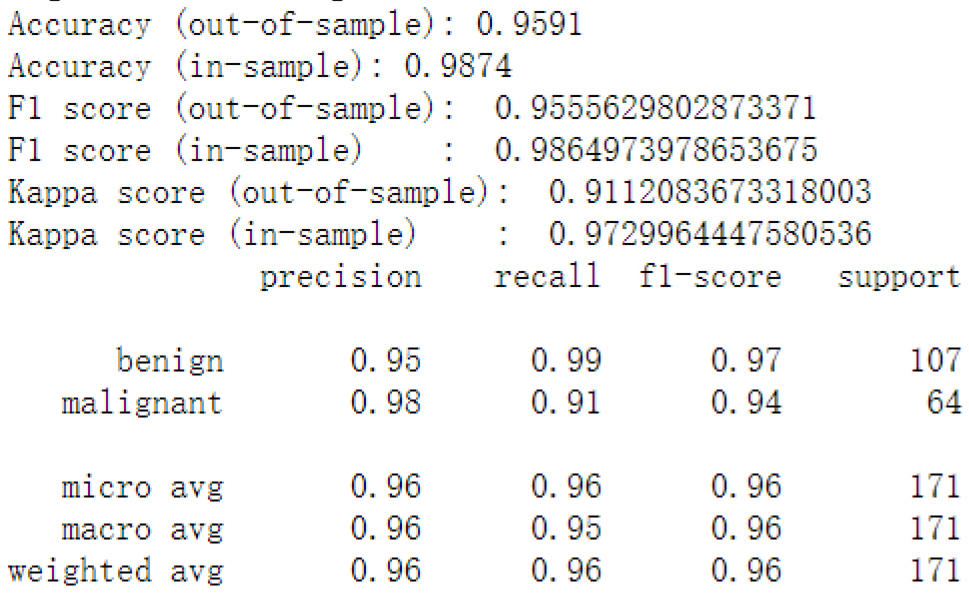
III. KNN Model

For my KNN model, I specify the number of neighbors to be 3 and the distance feature to be standard Euclidean.



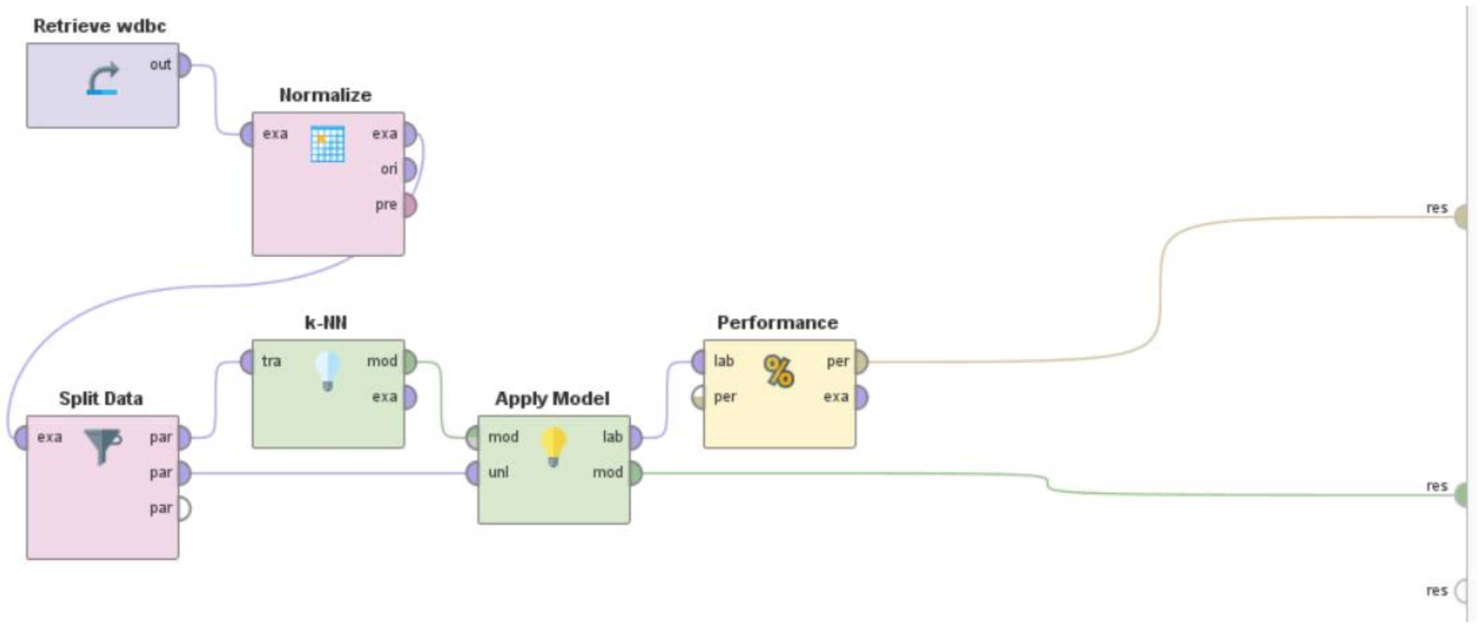
```
knn = neighbors.KNeighborsClassifier(n_neighbors=3,
                                     p=2,
                                     metric='minkowski') #The
                                                         # wit
knn = knn.fit(X_train_std, y_train)                      # wit
```

The performance of this model is relatively good. It has an accuracy of 96% on the test set, and both its precision and recall for benign records and malignant records are over 90%. F-measure is 97% for benign and 94% for malignant.

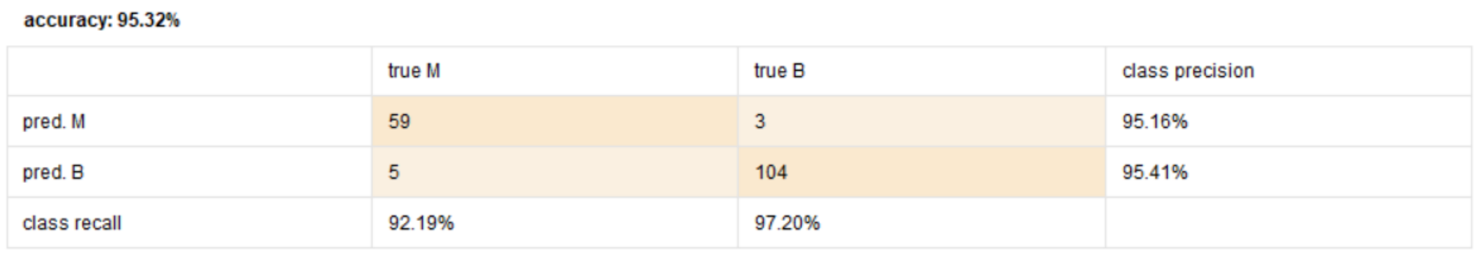


```
Accuracy (out-of-sample): 0.9591
Accuracy (in-sample): 0.9874
F1 score (out-of-sample):  0.9555629802873371
F1 score (in-sample)    :  0.9864973978653675
Kappa score (out-of-sample):  0.9112083673318003
Kappa score (in-sample)    :  0.9729964447580536
              precision    recall  f1-score   support

      benign       0.95      0.99      0.97       107
   malignant       0.98      0.91      0.94        64

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

In RapidMiner, my process for the KNN model looks like this:



The accuracy of my normalized KNN model in RapidMiner is 95.32%.



accuracy: 95.32%

	true M	true B	class precision
pred. M	59	3	95.16%
pred. B	5	104	95.41%
class recall	92.19%	97.20%	

IV. Logistic Regression Model

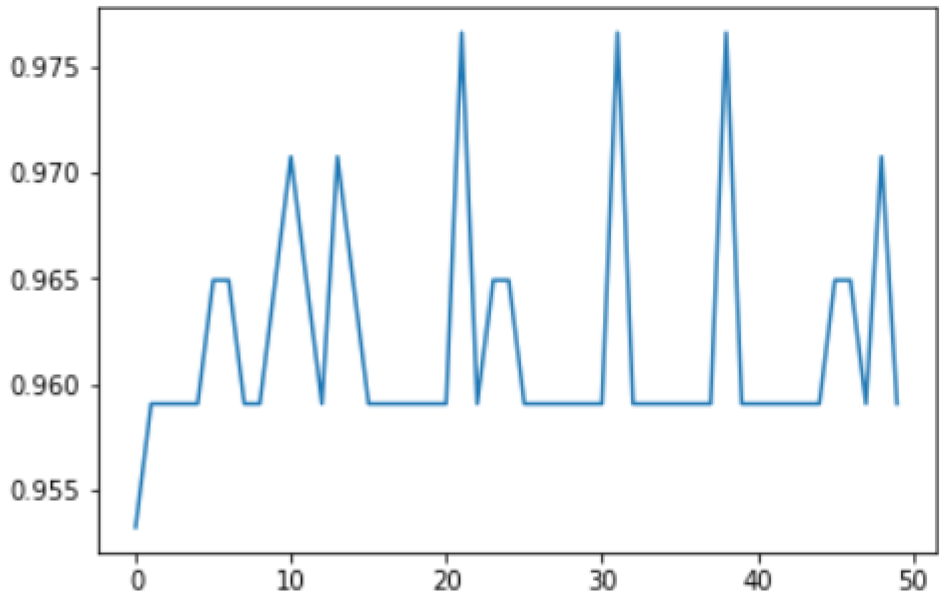
To determine the proper value for my lambda, I run a FOR loop to find the C value that yields the highest out-of-sample accuracy.

```
a = []
for i in range(50):
    clf = linear_model.LogisticRegression(C=10**i)
    clf = clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    y_pred_insample = clf.predict(X_train)
    a.append(accuracy_score(y_test, y_pred))
```

As demonstrated in the chart, the highest out-of-sample is 97.66% and is yielded at C= 10^21. Thus, I launch a logistic regression model with C= 10^21.

```
plt.plot(a)
print(np.max(a))
print(np.where(a == np.max(a)))
```

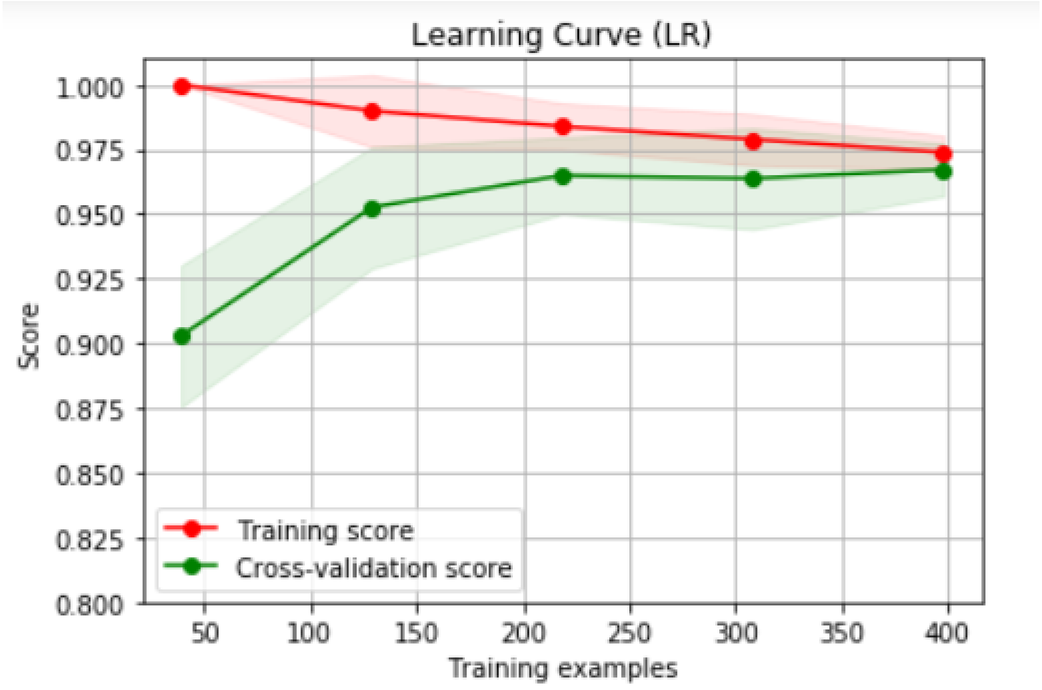
```
0.9766081871345029
(array([21, 31, 38], dtype=int64),)
```

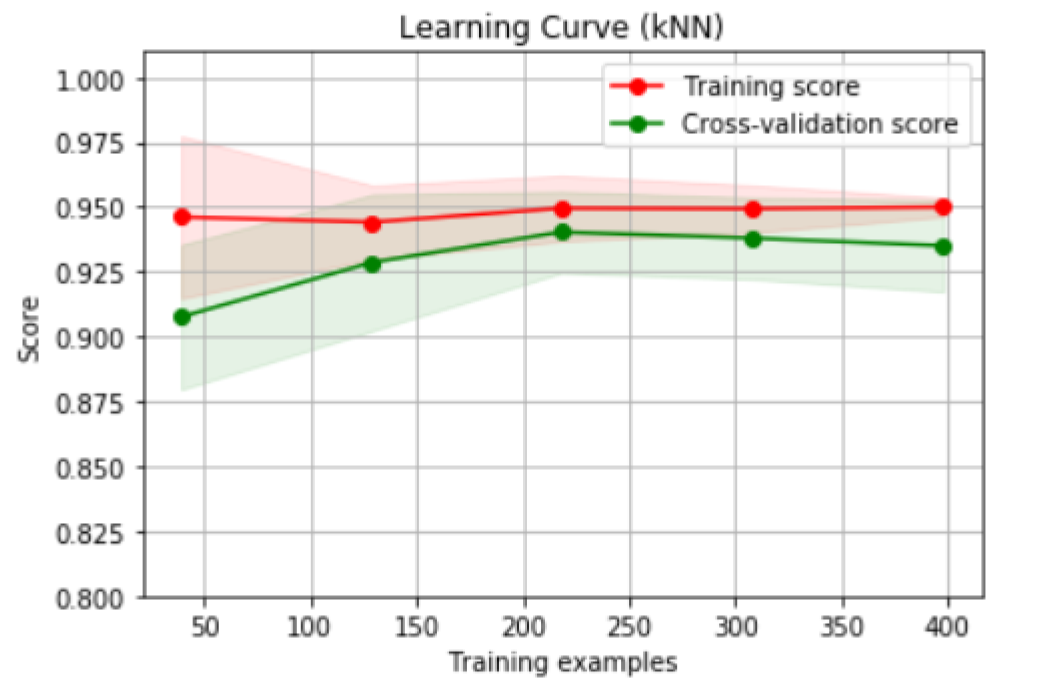


```
clf = linear_model.LogisticRegression(C=10**21)
clf = clf.fit(X_train, y_train)
```

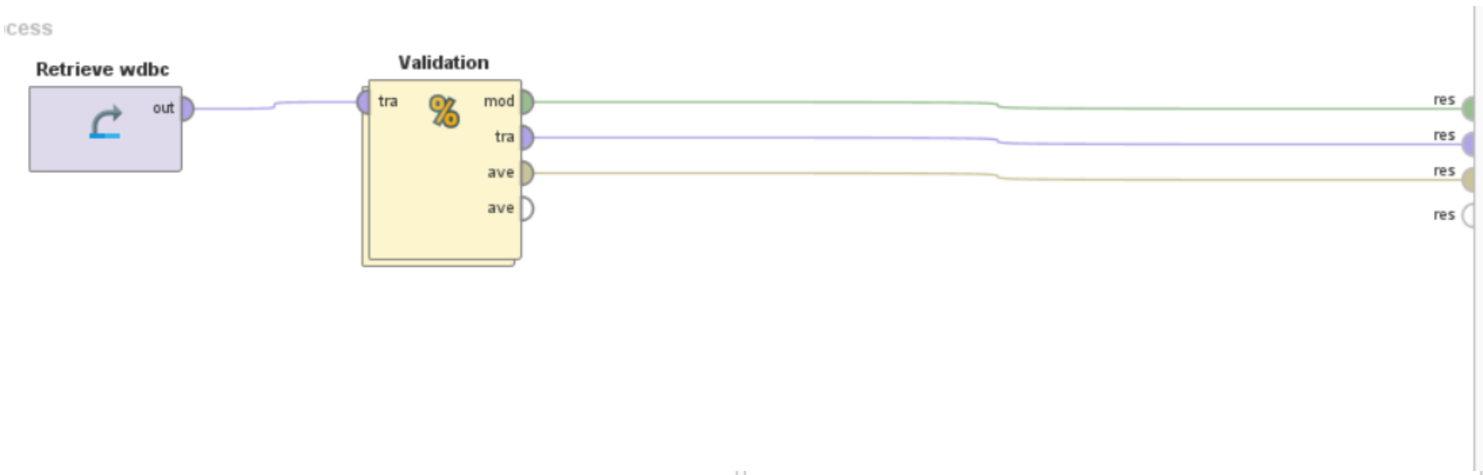
The LR model has an accuracy of 97.66% for test set. Both precisions and recalls of LR model are higher than the KNN model, showing that LR model is probably a better choice for this dataset. This can also be concluded from the learning curves.

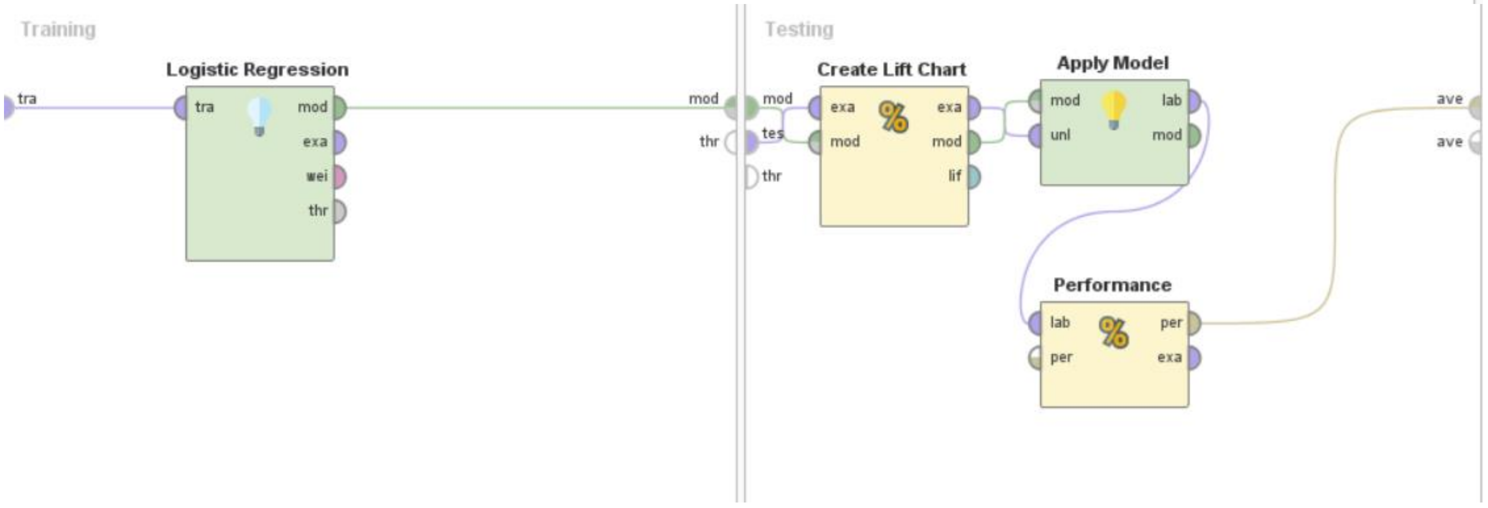
```
Accuracy (out-of-sample): 0.9766
Accuracy (in-sample): 0.9950
0.0 [9.99981366e-01 1.86337676e-05] 1.0
              precision    recall  f1-score   support

      benign       0.97      0.99      0.98       107
   malignant       0.98      0.95      0.97        64

   micro avg       0.98      0.98      0.98       171
   macro avg       0.98      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171
```



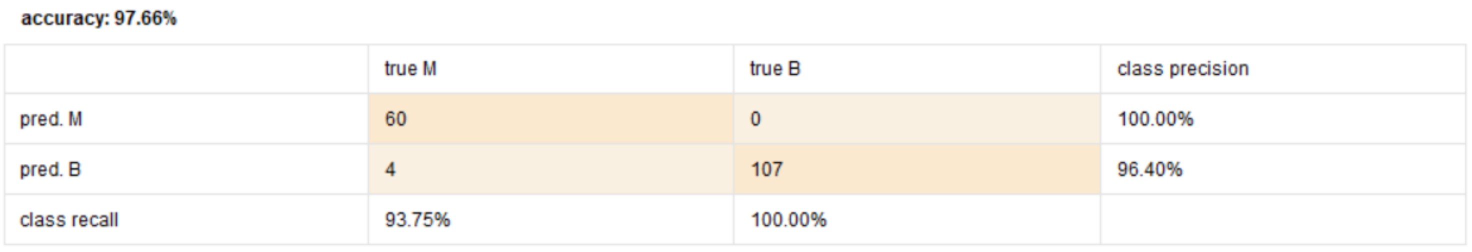


In RapidMiner, my LR model looks like this:





And the results are:



accuracy: 97.66%

	true M	true B	class precision
pred. M	60	0	100.00%
pred. B	4	107	96.40%
class recall	93.75%	100.00%	

Multicollinearity is not a big problem here for both models, because the results are great and show the potential multicollinearity doesn’t hurt the model much in reality.

From the confusion matrices and learning curves, we can conclude that the LR model is better for this dataset.

V. Cross Validation

```
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

# Fit model to all the data
clf_lr = linear_model.LogisticRegression(C=10**21)

# Evaluate performance with cross-validation
# Read more about cross_val_score in the following link
# http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cros

# Accuracy
scores=cross_val_score(clf_lr, X, y, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print(scores)

# F-1 scores
scores_f1=cross_val_score(clf_lr, X, y, cv=5, scoring='f1_macro')
print("F1-score: %0.2f (+/- %0.2f)" % (scores_f1.mean(), scores_f1.std() * 2))
print(scores_f1)
```

```
Accuracy: 0.95 (+/- 0.02)
[0.94782609 0.93913043 0.96460177 0.94690265 0.96460177]
F1-score: 0.95 (+/- 0.02)
[0.94428295 0.9340218  0.96172087 0.94368771 0.96245847]
```

Appendix (Data Description)

1.Title: Wisconsin Diagnostic Breast Cancer (WDBC)

Results:

- predicting field 2, diagnosis: B = benign, M = malignant

2. Number of instances: 569

3. Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)

4. Attribute information

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

5. Missing attribute values: none

6. Class distribution: 357 benign, 212 malignant