



Homework #2 Part 2

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

Note: This is a team homework assignment. Discussing this homework with your classmates outside your MSBA team 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 ____70____ points

ATTENTION: HW2 has two parts. Please first complete the Quiz “HW2_Part1” on Canvas. Then, proceed with Part 2 in the following page. You will need to submit (a) a PDF file with your answers and screenshots of Python code snippets as well as Rapidminer repositories and (b) the Python code and Rapidminer repositories. (70 points) [Mining publicly available data. Please implement the following models with both Rapidminer and 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> and is also provided as in the appendix of this homework assignment.

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 B or malignant M) .

50 Points (Python):

a) (10 points) Load the data. Then, explore the data by reporting summary statistics and a correlation matrix. Show your code.

a. Summary Statistics of the breast-cancer-wisconsin data:

After load the data, we assigned the column name based on given information with the following code:

```
# Define the base features
base_features = ["radius", "texture", "perimeter", "area", "smoothness", "compactness", "concavity", "concave points",
                "symmetry", "fractal dimension"]

# Define the computations
computations = ["Mean", "SE", "Worst"]

# Create the header list
headers = ["ID number", "Diagnosis"]

# Append computed features to the headers
for computation in computations:
    for feature in base_features:
        headers.append(f"{computation} {feature}")

headers

df = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data",
                 header=None, names=headers)
df.head()
```

The table below is the corresponding outputs.

	ID number	Mean radius	Mean texture	Mean perimeter	Mean area	Mean smoothness	Mean compactness	Mean concavity	mean concave points	Mean symmetry	...	Worst radius	Wo text
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.0000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	16.269190	25.6772
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	4.833242	6.1462
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	7.930000	12.0200
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	13.010000	21.0800
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	14.970000	25.4100
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	18.790000	29.7200
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	36.040000	49.5400

b. Correlation matrix of the breast-cancer-wisconsin data:

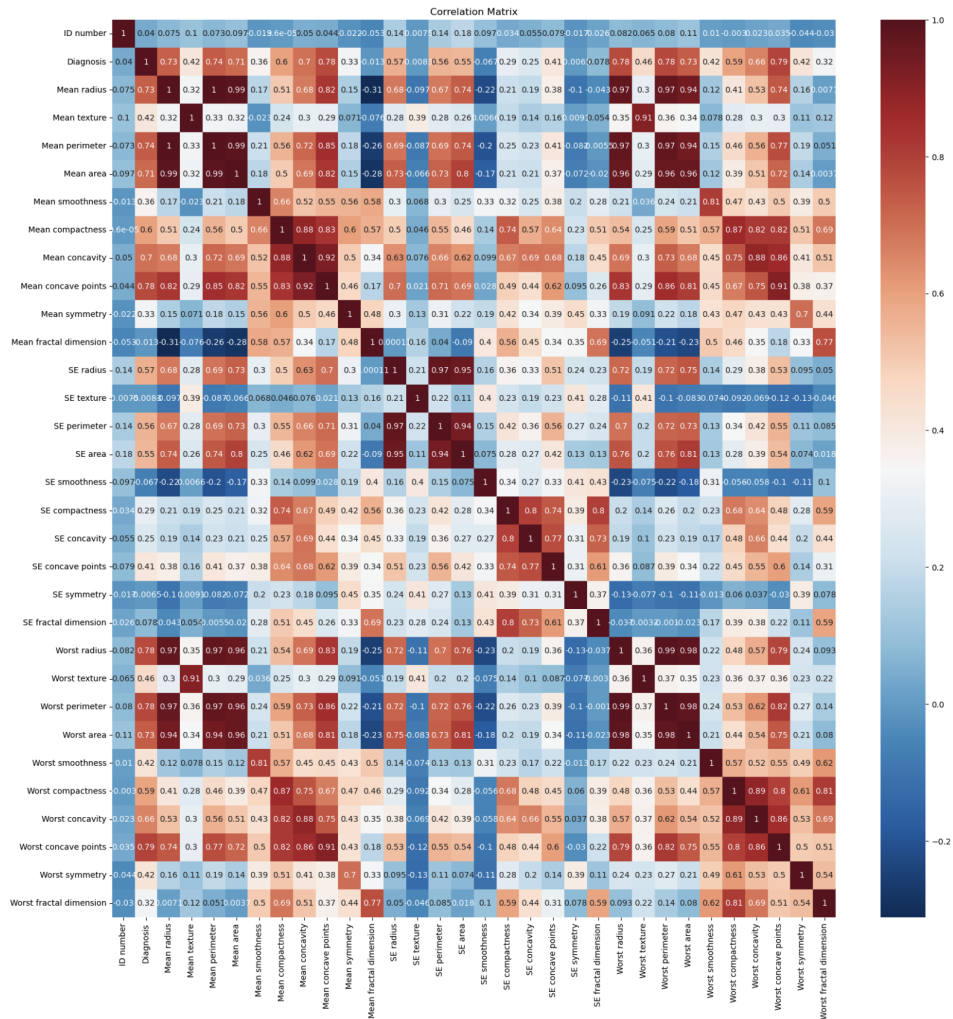
To provide a clearer understanding of the correlation matrix, we created the heatmaps. The corresponding codes and outputs are as below:

```
summary_stats = df.describe()
print("Summary Statistics:")
print(summary_stats)

correlation_matrix = df.iloc[:,2:].corr() # Calculate the correlation matrix
correlation_matrix

plt.figure(figsize=(20, 20))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix Heatmap")
plt.show()
```

Heatmap Results:



- b) (12 points) Perform a predictive modeling analysis on this dataset to predict the type (benign B or malignant M) using a k-NN technique (for k=3) and the Logistic Regression technique. Please be specific about what other parameters you specified for your models. Briefly discuss your modeling process (e.g., validation technique, any preprocessing steps, parameters used to build the models, etc.) and show your code. Report the estimated coefficients of the Logistic Regression technique.

Validation Process:

1. Check Correlation with Target Variable:

After building the heated map to demonstrate the **correlation** between attributes and our target variable Diagnosis, we have found some attributes that are highly correlated. When independent variables are highly correlated, multicollinearity will be a problem. This can lead to unstable coefficient estimates,

making it difficult to determine the individual impact of predictors on the response. As shown in the following bar graph, if we determine certain thresholds, the variables with higher correlations could be eliminated or perform regularizations, including either L1 Lasso regression or L2 ridge. For convenience, we suspected to use a threshold of 0.7 as it is a conventional choice.

```
import matplotlib.pyplot as plt
import seaborn as sns

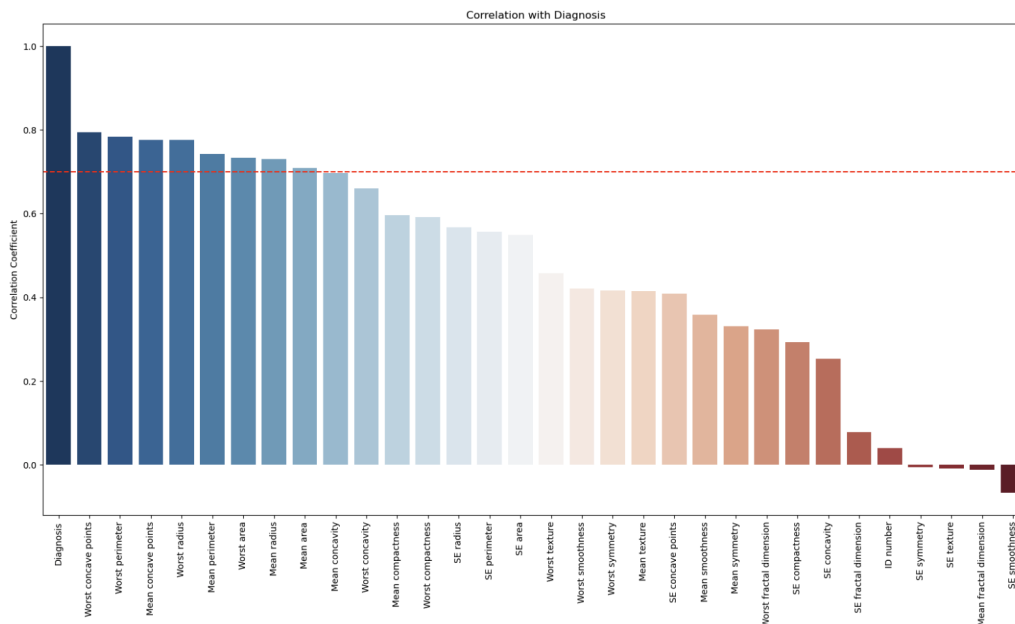
# Compute correlation of all columns with each other
full_correlation_matrix = df.corr()

# Select the 'Diagnosis' row or column from the full correlation matrix
correlation_with_diagnosis = full_correlation_matrix['Diagnosis'].sort_values(ascending=False)

# Plotting
plt.figure(figsize=(20, 10))
sns.barplot(x=correlation_with_diagnosis.index, y=correlation_with_diagnosis.values, palette="RdBu_r")

# Add a horizontal line at y=0.7
plt.axhline(y=0.7, color='r', linestyle='--')

plt.title("Correlation with Diagnosis")
plt.xticks(rotation=90)
plt.ylabel('Correlation Coefficient')
plt.show()
```



The screenshot demonstrates how we perform the validation for eliminating attributes that are highly correlated with our target variable.

```
correlation_with_diagnosis = df.corr()['Diagnosis']
columns_to_drop = correlation_with_diagnosis[correlation_with_diagnosis.abs() > 0.7].index.drop('Diagnosis')
df.drop(columns_to_drop, axis=1, inplace=True)
```

2. Detect Missing Values:

During the process, we performed a thorough evaluation of our datasets, paying close attention to any potential **missing values** in our datasets. This helps guarantee the completeness and accuracy of our dataset, consequently avoiding mistakes in any future analysis or modeling using the dataset.

After performing a thorough evaluation of our datasets, we can see, as shown in the table, there is no missing value included in our given dataset, which means our data is in a good position to be used in our models.

```
# 1.1 Handling missing values
# For this dataset, there aren't any missing values. But let's check to be sure.
missing_values = df.isnull().sum()
```

```
missing_values
ID number      0
Diagnosis      0
Mean radius    0
Mean texture    0
Mean perimeter  0
Mean area      0
Mean smoothness 0
Mean compactness 0
Mean concavity  0
Mean concave points 0
Mean symmetry   0
Mean fractal dimension 0
SE radius       0
SE texture      0
SE perimeter    0
SE area         0
SE smoothness   0
SE compactness  0
SE concavity    0
SE concave points 0
SE symmetry     0
SE fractal dimension 0
Worst radius    0
Worst texture   0
Worst perimeter 0
Worst area      0
Worst smoothness 0
Worst compactness 0
Worst concavity 0
Worst concave points 0
Worst symmetry  0
Worst fractal dimension 0
dtype: int64
```

Parameters used:

For the k-NN model, the following parameters were used:

- n_neighbors: 3 (This means the model looks at the three nearest points to make a prediction.)
- metric: Minkowski (This is a generalization of the Euclidean and Manhattan distances.)
- p: 2 (With p=2, the Minkowski metric becomes the Euclidean distance.)
- weights: Uniform (This means all points in the neighborhood are weighted equally.)

For the Logistic Regression model, we used:

- max_iter: 10000 (This is the maximum number of iterations for the solvers to converge.)
- random_state: 42 (For reproducibility.)

Estimated Coefficients of the Logistic Regression:

The weights of the attributes are: [[-3.52934663 -0.09482721 0.58631845 -0.0246688 0.13048482
0.64178528

0.8910687 0.37953675 0.17469563 0.04902986 -0.11077869 -1.58405815
0.13606778 0.11972166 0.01855867 0.13044177 0.18841705 0.05151257
0.06434593 0.01356018 -3.33077087 0.30646393 0.12944361 0.05161542
0.23831075 1.87744164 2.36375563 0.69796482 0.70934228 0.19373244]]

The weights of the intercepts are: [-0.58020583]

- c) (13 points) Compare the generalization performance of the k-NN model with the Logistic Regression model. Make sure you report the confusion matrix, the predictive accuracy, precision, recall, and f-measure. Briefly discuss the results and show your code.

k-NN performance:

```
Accuracy (out-of-sample): 0.96
Accuracy (in-sample): 0.99
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

      B         0.95         0.99         0.97         107
      M         0.98         0.91         0.94          64

   accuracy              0.96         171
  macro avg         0.96         0.95         0.96         171
 weighted avg         0.96         0.96         0.96         171
```

Logistic Regression:

```
Accuracy (in-sample): 0.99
Accuracy (out-of-sample): 0.97
F1 score (in-sample): 0.9821428571
F1 score (out-of-sample): 0.9647058824
Kappa score (in-sample): 0.9716904826
Kappa score (out-of-sample): 0.9437314906
```

Detailed Classification Report (out-of-sample):

```
      precision    recall  f1-score   support

      B         0.97         0.99         0.98         71
      M         0.98         0.95         0.96         43

   accuracy              0.97         114
  macro avg         0.97         0.97         0.97         114
 weighted avg         0.97         0.97         0.97         114
```

Accuracy rate is the proportion of correct diagnostics in the dataset. In our example, 97% of out-of-sample instances are correctly classified by the logistic regression model, and 96% of out-of-sample instances are correctly classified by the KNN model. Both the KNN model and the logistic regression model have good accuracy rates, while logistic regression performs better in the out-of-sample accuracy. In addition, both models are not overfitting in this case since the difference in model accuracy between the in-sample and out-of-sample is minor.

Recall (malignant) in the mathematical perspective, $TP/(TP + FN)$, provides the proportion of actual positives that were correctly classified by the model; Recall (benign) in the mathematical perspective, $TN/(TN + FP)$, provides the proportion of actual negatives that were correctly classified by the model; The higher recall score indicates that the model classified most of the positive(negative) samples correctly.

According to the outcome in the K-NN model, a 99% recall for the "benign" classification means that 99% of the actual benign samples were correctly identified and classified as benign by the model, and only 1% of the benign samples were missed or incorrectly classified as malignant. A 91% recall for the "malignant" classification means that 91% of the actual malignant samples were correctly identified and classified as

malignant by the model, and only 9% of the malignant samples were missed or incorrectly classified as benign.

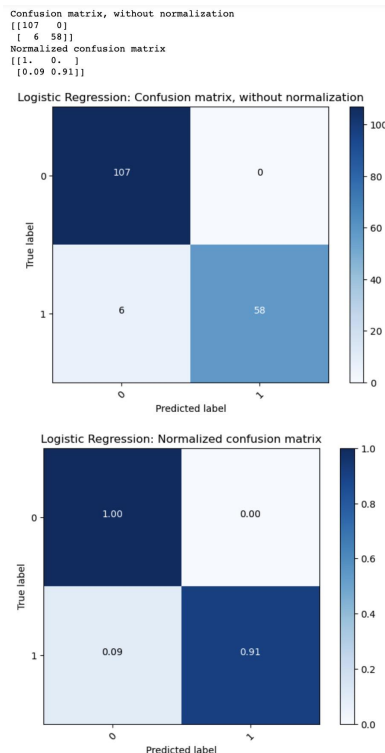
As for the logistic model, a 99% recall for the "benign" classification means that 99% of the actual benign samples were correctly identified and classified as benign by the model, and only 1% of the benign samples were missed or incorrectly classified as malignant. A 95% recall for the "malignant" classification means that 95% of the actual malignant samples were correctly identified and classified as malignant by the model, and only 5% of the malignant samples were missed or incorrectly classified as benign.

As a result, the logistic regression model performs better in the out-of-sample recall metric.

Precision (positive) measures how many of the positive predictions made by a classification model are actually positive. It's a ratio of correctly predicted positive observations to the total predicted positives. Both the KNN model and logistic regression model have a precision (malignant) of 98%. In other words, when the model predicts a patient's cancer is malignant, there's a 98% chance that the cancer is truly being malignant. The KNN model has a precision (benign) of 95%. In other words, when the model predicts a patient's cancer is benign, there's a 95% chance that the cancer is truly benign. The logistic regression model has a precision (benign) of 97%. In other words, when the model predicts a patient's cancer is benign, there's a 97% chance that the cancer is truly benign. In our example, the logistic regression model performs better in the out-of-sample precision metric.

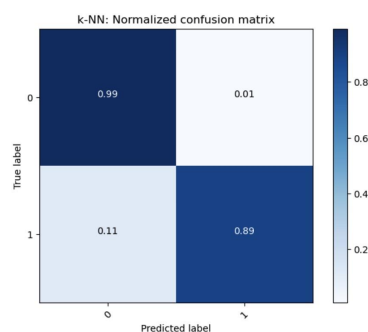
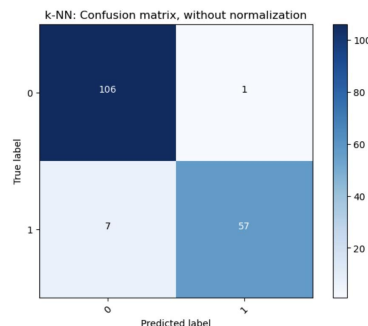
F1 score integrates both recall and precision. KNN has a 95% out-of-sample F1 score, and logistic regression has a 96% out-of-sample F1 score, so logistic regression performs slightly better than the K-NN model in terms of out-of-sample F1 score in terms of achieving a harmonious balance between precision and recall on unseen data.

Confusion Matrix:



In the **Logistic Regression model**, 100% of instances predicted are True Positive, 0% of instances predicted are False Positive, 9% of instances predicted are False Negative, and 91% of instances predicted are True Negative.


```
Confusion matrix, without normalization
[[106  1]
 [ 7 57]]
Normalized confusion matrix
[[0.99 0.01]
 [0.11 0.89]]
```



In the **KNN model**, 99% of instances predicted are True Positive, 1% of instances predicted are False Positive, 11% of instances predicted are False Negative, and 89% of instances predicted are True Negative.

- d) (15 points) What generalization performance metric would you prefer to use in order to choose the best performing model in this context and why? Please be clear about any assumptions you might make when you choose the generalization performance metric you would prefer.

In this context, we prioritize using recall to select the best-performing model.

While we are dealing with a cancer diagnosis, if our model consistently under-identifies actual cancer cases, it could lead to a significant number of patients not receiving timely treatment, with potentially life-threatening outcomes.

Missing a positive diagnosis (false negative) would be far more costly than a false positive. A low recall would mean the model has missed many positive cases, resulting in a high volume of false negatives, so we want to maximize the recall of a model. When cancer is actually malignant but diagnosed as benign, the consequences can be fatal. Given the high costs associated with such misdiagnoses, it's imperative that we emphasize **recall** as the generalization performance metric.

20 Points (Rapidminer):

Perform a predictive modeling analysis on this dataset to predict the type (benign B or malignant M) using a k-NN technique (for k=3) and the Logistic Regression technique. Compare the generalization performance of the k-NN model with the Logistic Regression model. Make sure you report the confusion matrix, the predictive accuracy, precision, recall, and f-measure.

- a) [20 points] Please show below screenshots of the models you have built using Rapidminer, the results, and the parameters you have specified.
 - a. [8 points] Data Preview

i.

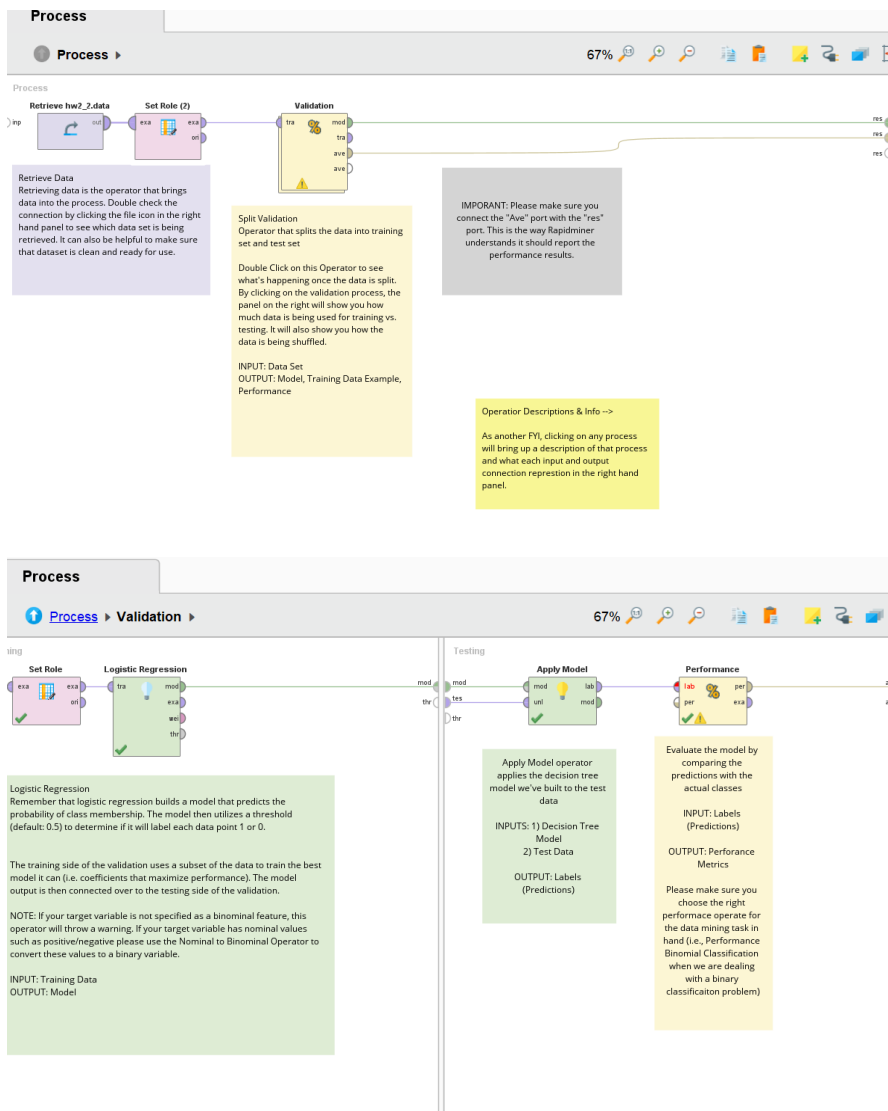
Name	Type	Missing	Statistics	Filter (32 / 32 attributes)
att1	Integer	0	Min: 8670, Max: 911320502, Average: 30371831.432	
att2	Nominal	0	Least: M (212), Most: B (357), Values: B (357), M (212)	
att3	Real	0	Min: 6.981, Max: 28.110, Average: 14.127	
att4	Real	0	Min: 9.710, Max: 39.280, Average: 19.290	
att5	Real	0	Min: 43.790, Max: 188.500, Average: 91.969	
att6	Real	0	Min: 143.500, Max: 2501, Average: 654.889	
att7	Real	0	Min: 0.053, Max: 0.163, Average: 0.096	

Showing attributes 1 - 32 Examples: 569 Special Attributes: 0 Regular Attributes: 32

b. [8 points] Logistic Regression

i. Screenshots for Logistic Regression Model Setup (Rapidminer Processes)

(Insert Screenshots here – 2 screenshots are expected here; one for the upper layer and one inside the validation technique)



ii. Screenshot for Logistic Regression Performance

(Insert Screenshot here)

Accuracy

Table ViewPlot View

accuracy: 93.57%

	true M	true B	class precision
pred. M	71	9	88.75%
pred. B	2	89	97.80%
class recall	97.26%	90.82%	

Precision

Table ViewPlot View

precision: 97.80% (positive class: B)

	true M	true B	class precision
pred. M	71	9	88.75%
pred. B	2	89	97.80%
class recall	97.26%	90.82%	

Recall

Table ViewPlot View

recall: 90.82% (positive class: B)

	true M	true B	class precision
pred. M	71	9	88.75%
pred. B	2	89	97.80%
class recall	97.26%	90.82%	

F1 Measure

Table ViewPlot View

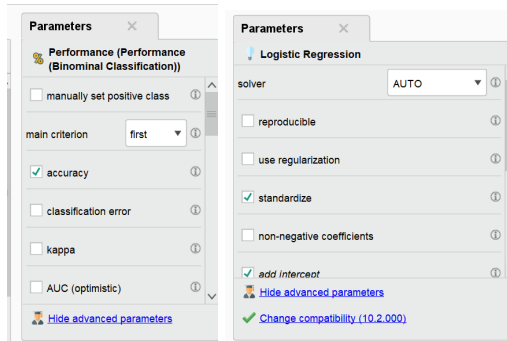
f_measure: 94.18% (positive class: B)

	true M	true B	class precision
pred. M	71	9	88.75%
pred. B	2	89	97.80%
class recall	97.26%	90.82%	

iii. Screenshot for Logistic Regression Results (Coefficients)
(Insert Shreenshot here)

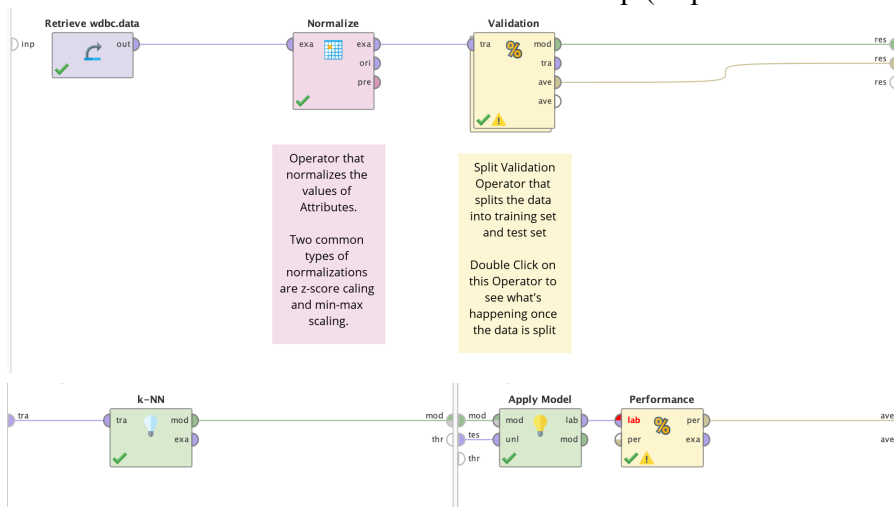
Attribute	Coefficient				
att1	0.000	att16	-11.486		
att3	336.976	att17	-1791.856		
att4	-6.534	att18	-6098.389	att26	1.096
att5	4.266	att19	5211.568	att27	1561.472
att6	-3.550	att20	-22867.640	att28	534.299
att7	-4144.690	att21	10288.291	att29	-506.592
att8	3285.021	att22	57312.933	att30	-497.763
att9	-1728.287	att23	-113.308	att31	-1779.926
att10	-2281.412	att24	-7.845	att32	-5162.489
att11	1580.948	att25	-6.395	Intercept	-5.372
att12	-2736.834				
att13	15.275				
att14	41.495				
att15	72.582				

- iv. Screenshot for Logistic Regression Rapidminer Operator Parameters (click on Logistic Regression operator and then take a screenshot of the Parameters window on the right)
(Insert Screenshot here)



c. [8 points] kNN

i. Screenshots for kNN Model Setup (Rapidminer Processes)



ii. Screenshot for kNN Performance

accuracy: 98.83%

	true M	true B	class precision
pred. M	72	1	98.63%
pred. B	1	97	98.98%
class recall	98.63%	98.98%	


precision: 98.98% (positive class: B)

	true M	true B	class precision
pred. M	72	1	98.63%
pred. B	1	97	98.98%
class recall	98.63%	98.98%	

recall: 98.98% (positive class: B)			
	true M	true B	class precision
pred. M	72	1	98.63%
pred. B	1	97	98.98%
class recall	98.63%	98.98%	

f_measure: 98.98% (positive class: B)			
	true M	true B	class precision
pred. M	72	1	98.63%
pred. B	1	97	98.98%
class recall	98.63%	98.98%	

- iii. Screenshot for kNN Rapidminer Operator Parameters (click on kNN operator and then take a screenshot of the Parameters windows on the right)


Normalize

attribute filter type

all

☐ invert selection

☐ include special attributes


min

0.0

max

1.0

Parameters


Validation (Split Validation)

split


relative

split ratio

0.7

sampling type

shuffled sampling


k-NN

k

3

☒ weighted vote

measure types

MixedMeasures

mixed measure

MixedEuclideanDistance

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 ($\text{perimeter}^2 / \text{area} - 1.0$)

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