DATA MINING REPORT

I. Abstract/Task Definition

The m-Health industry has been on the rise during this pandemic. m-Health applications enable clients to communicate their illnesses with doctors online. Our team developed several multilabel classification models from given texts consisting of Doctoral Specialisations Label on Medical Questionnaires. These models came from a classifier, SVM, Naive Bayes, Ensemble Learning, and Problem Transformation. Results from these models varied and each had their own strengths, but adaboost stood out the most as it produced the highest accuracy.

II. Methodology

A. Pre-Processing

In this stage, we process the raw data into vectorized data that is ready to use for model training and testing. First, we transform the XML format into regular CSV format by using a software converter. Then we apply data cleaning by using Sastrawi, and finally we vectorize the data by using both word2vec and bag-of-words. The changes made to the tables were adding the dataset human into the machine dataset and adding NaN into the extra columns, and concatenating “judul” with “isi” into the “isi” column as they both contain texts with keywords that would be cleaned later.

B. Model Training/Feature Engineering

1. Text Vectorization

a) Word2vec

Word2vec is a natural language processing model that uses a neural network model to learn word associations from a large corpus of text. Once trained, the model can detect words with high similarity or synonymity, and even suggest additional words for a partial sentence.

b) BoW

Bag-of-Words model is a simplified representation of a text, by using a multiset of its words (called a bag), disregarding grammar and even word order but keeping the multiplicity.

2. Supervised
    a) Linear Support Vector Machine

        The first model made was Linear SVM using word2vec vectorization. The Linear SVM model was used as a starting point for familiarity as it was introduced in the module given for this task. Linear SVM is a supervised learning method that uses training data from the machine dataset that has been combined with the human dataset. Word2Vec was used.

    b) Naive Bayes: Gaussian

        Gaussian NB is a variation of the Naive Bayes Classifiers that supports continuous valued features and models each as conforming to a Gaussian (normal) distribution. This model is good for classifying data that conforms to a Gaussian distribution, even with some variance. Word2Vec was used.

    c) Naive Bayes: Multinomial

        Gaussian NB is a variation of the Naive Bayes Classifiers that are designed to handle texts/documents using word counts as its underlying method of calculating probability. This model works best for handling classification that involve simple classes that do not involve sentiment analysis. BoW was used.

    d) MLP

        MLP classifier is a form of supervised learning using backpropagation. MLPClassifier() was used for creating a model from the machine dataset, and was then used for the testing dataset that has been cleaned. BoW was used to vectorize the “isi” column.

3. Techniques

    There are multiple techniques to solve multi label classification problems. In this experiment, we used the Problem Transformation and Ensemble Learning techniques with text vectorization using Word2vec.

    a) Problem Transformation

        Problem transformation is the technique that transforms multi-label problems into single-label problems. There are three ways to carry out problem transformation: Binary Relevance,

Classifier Chains, and Label Powerset. In this experiment, we used the implementation provided by the skmultilearn library and Gaussian Naive Bayes as the base classifier.

(1) Binary Relevance

This technique simply treats each label as its own classification problem. After the dataset is pre-processed and vectorized, we then plug in the training data and training label into the library.

```
# Binary Relevance
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB

# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(GaussianNB())

# train
classifier.fit(X_train_w2v, y_train_encoded.astype(int))

# predict
predictions = classifier.predict(X_test_w2v)
```

Since this technique is the simplest among the others, it is expected to not perform excellently.

(2) Classifier Chains

This technique is pretty similar to Binary Relevance, but it preserves label correlation by training each next classifier on the input space and previous classifiers, forming a chain.

```
# using Classifier Chains
from skmultilearn.problem_transform import ClassifierChain
from sklearn.naive_bayes import GaussianNB

# initialize classifier chains multi-label classifier
# with a gaussian naive bayes base classifier
cc_classifier = ClassifierChain(GaussianNB())

# train
cc_classifier.fit(X_train_w2v, y_train_encoded.astype(int))

# predict
pred_cc = cc_classifier.predict(X_test_w2v)
```

(3) Label Powerset

This technique transforms the problem into a multi-class problem by training the classifier on the

unique label combinations in the dataset. It marks the data with the same set of labels into a single problem. For example:

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

```
# using Label Powerset
from skmultilearn.problem_transform import LabelPowerset
from sklearn.naive_bayes import GaussianNB

# initialize Label Powerset multi-label classifier
# with a gaussian naive bayes base classifier
lp_classifier = LabelPowerset(GaussianNB())

# train
lp_classifier.fit(X_train_w2v, y_train_encoded.astype(int))

# predict
pred_lp = lp_classifier.predict(X_test_w2v)
```

b) Ensemble Learning

Ensemble learning is a method where you use multiple models and combine them to process better results. The idea is that single models are usually weak on their own, but combined with other models and enhancements, better results should be achievable. Some ensemble learning techniques include bagging and boosting. However, in this experiment we will only touch on boosting.

Boosting utilises algorithms to convert weak learner models into strong learners. Boosting involves giving higher weights to misclassified predictions from previous learners in order to reduce the errors going forward until the sequence arrives at the final and strongest models.

In this experiment, we separated each target label to build its own boosting model which is then used to make predictions for that label. The boosting methods used in this experiment are Adaboost and XGboost.

(1) Adaboost (Adaptive Boost)

```
# Membagi dataset yang ada menjadi 2 bagian, train serta test
X_train_ens, X_test_ens, Y_train_ens, Y_test_ens = train_test_split(X_train_w2v, y[14], test_size = 0.3, random_state = 100)
scaler = StandardScaler()
X_train_ens = scaler.fit_transform(X_train_ens)
X_test_ens = scaler.transform(X_test_ens)

adaboost_paru = AdaBoostClassifier(n_estimators = 50, learning_rate = 1.0).fit(X_train_ens, Y_train_ens)

pred_res_paru = adaboost_paru.predict(X_test_ens)
```

(2) XGboost

```
[ ] # Membagi dataset yang ada menjadi 2 bagian, train serta test
    X_train_ens, X_test_ens, Y_train_ens, Y_test_ens = train_test_split(X_train_w2v, y[0], test_size = 0.3, random_state = 100)
    scaler = StandardScaler()
    X_train_ens = scaler.fit_transform(X_train_ens)
    X_test_ens = scaler.transform(X_test_ens)
    xgboost_bidan = XGBClassifier(n_estimators = 100, learning_rate = 0.1).fit(X_train_ens, Y_train_ens)
    pred_res_bidan = xgboost_bidan.predict(X_test_ens)
```

III. Discussion

A. Model Evaluation Score

1. Problem Transformation

a) Binary Relevance

.:Informasi:.

Uploaded File: binary_relevance_preds.csv
Type: text/csv
Size: 116.10546875 KB
EMR: 11.069354287667
Hamming Loss: 26.157328322515
Precision Macro: 26.630504189311
Recall Macro: 74.566294851579
F1-score Macro: 36.232784355685
Precision Micro: 23.183486874004
Recall Micro: 74.262258708675
F1-score Micro: 35.335726351351

b) Classifier Chains

.:Informasi:.

Uploaded File: classifier_chains2.csv
Type: text/csv
Size: 116.10546875 KB
EMR: 13.221728732491
Hamming Loss: 21.015544926546
Precision Macro: 29.264466568354
Recall Macro: 60.03376421205
F1-score Macro: 36.380461498102
Precision Micro: 26.299422478898
Recall Micro: 65.675615708897
F1-score Micro: 37.558685446009

c) Label Powerset

> ## .:Informasi:.
>
> Uploaded File: label_powerset2.csv
> Type: text/csv
> Size: 116.10546875 KB
> **EMR**: 26.819268875982
> **Hamming Loss**: 11.165442432525
> **Precision Macro**: 46.139890418072
> **Recall Macro**: 57.760275047597
> **F1-score Macro**: 49.283259966639
> **Precision Micro**: 44.184922680412
> **Recall Micro**: 60.860883070779
> **F1-score Micro**: 51.199253383108

As expected, the problem transformation methods' performance go in order with Binary Relevance performing the worst, Classifier Chains performing better, and Label Powerset performing the best out of the three.

2. Ensemble Learning
    a) Adaboost

.:Informasi:.

Uploaded File: ensemble_adaboost2.csv
Type: text/csv
Size: 116.10546875 KB
EMR: 16.843184147591
Hamming Loss: 9.2821147933037
Precision Macro: 58.603064899387
Recall Macro: 23.697515906313
F1-score Macro: 28.213568424416
Precision Micro: 53.244120032441
Recall Micro: 29.132460616818
F1-score Micro: 37.659543955256

Surprisingly, ensemble learning using Adaboost performed terribly despite the theory's promise of excellent performance. Adaboost only outperformed the other models in the Hamming Loss score.

b) XGboost

Compared to XGboost, Adaboost is much better in the metric figures. Hamming loss of XGboost was much higher than Adaboost, and all other metrics were lower than Adaboost, whereas the EMR was higher than Adaboost.

3. Naive Bayes

a) Gaussian

Uploaded File: datakedua.csv
Type: text/csv
Size: 116.10546875 KB
EMR: 10.796036897848
Hamming Loss: 25.702511103519
Precision Macro: 25.965665452393
Recall Macro: 74.892029361819
F1-score Macro: 35.964852133883
Precision Micro: 23.530652418448
Recall Micro: 74.262258708675
F1-score Micro: 35.737547381346

*jika terjadi error terkait 'mysql', coba unggah sekali lagi.

See Current Rankings

b) MNB (multinomial)

```
Akurasi: 0.9149436090225563
F1 Macro: 0.7556313491579009
F1 Micro: 0.9149436090225563
Hamming Loss: 0.08505639097744361
Recall Macro: 0.7318673243407402
Recall Micro: 0.9149436090225563
Precision Macro: 0.7983939538045077
Precision Micro: 0.9149436090225563
```

MNB showed a higher accuracy figure than Gaussian. Similarly, MNB also showed higher F1 Macro, F1 Micro, and Precision Macro, Micro figures. In addition, Recall Macro was higher by Gaussian, whereas Recall Micro was higher with MNB.

4. SVM
   a) Linear

```
Akurasi: 0.9176908617698092
F1 Macro: 0.7635159143883427
F1 Micro: 0.9176908617698092
Hamming Loss: 0.08230913823019086
Recall Macro: 0.7312001343481207
Recall Micro: 0.9176908617698092
Precision Macro: 0.8253658106883388
Precision Micro: 0.9176908617698092
```

Linear SVM showed impressive metric figures. F1 macro and micro were at 76 and 91. Furthermore, the hamming loss is relatively small, at 8%. Other metrics like recall and precision are relatively high with recall micro at 91%, precision macro and micro at 82% and 91%, which makes Linear SVM as one of our best models.

5. Classifier
   a) MLP Classifier

Uploaded File: databowmlp (1).csv
Type: text/csv
Size: 116.10546875 KB
EMR: 38.91356337547
Hamming Loss: 6.2030235736249
Precision Macro: 79.68739796147
Recall Macro: 44.592990125214
F1-score Macro: 55.572196284074
Precision Micro: 77.170963364993
Recall Micro: 50.47703572221
F1-score Micro: 61.032863849765

The EMR for MLP is relatively lower than the other models. However, the hamming loss for MLP is lower than all the other models. Furthermore, The precision micro and macro are considered high at around 77% and 79%. F1 and Recall metrics are around the 50-70% figures which do not stand out from the other models.

B. Model Comparison with Gold Standard

Comparison of models to the gold standard data was done by comparing each column of csv created from the models to the gold standard and averaging the number of matches.

1. Problem Transformation
    a) Binary Relevance
       Average accuracy: 73.78%
    b) Classifier Chains
       Average accuracy: 78.98%
    c) Label Powerset
       Average accuracy: 89.58%
2. Ensemble Learning
    a) Adaboost
       Average accuracy: 90.68%
    b) XGboost
       Average accuracy: 89.49%
3. Naive Bayes
    a) Gaussian NB
       Average accuracy: 67.54%
    b) MNB
       Average accuracy: 86.51%
4. SVM
    a) Linear
       Average accuracy: 84.76%
5. Classifier
    a) MLP Classifier

Average accuracy: 84.41%

C. Model Ranking Based on Results

By rank, when considering other metrics like F1, hamming loss, Precision, Recall, and EMR, the highest rank was achieved using the Multinomial Naive Bayes model. However, considering only the average accuracy to the gold standard in part B, ensemble learning using Adaboost had the highest number of matches to the gold standard with an average accuracy of 90.68%.