Assignment 1 Report

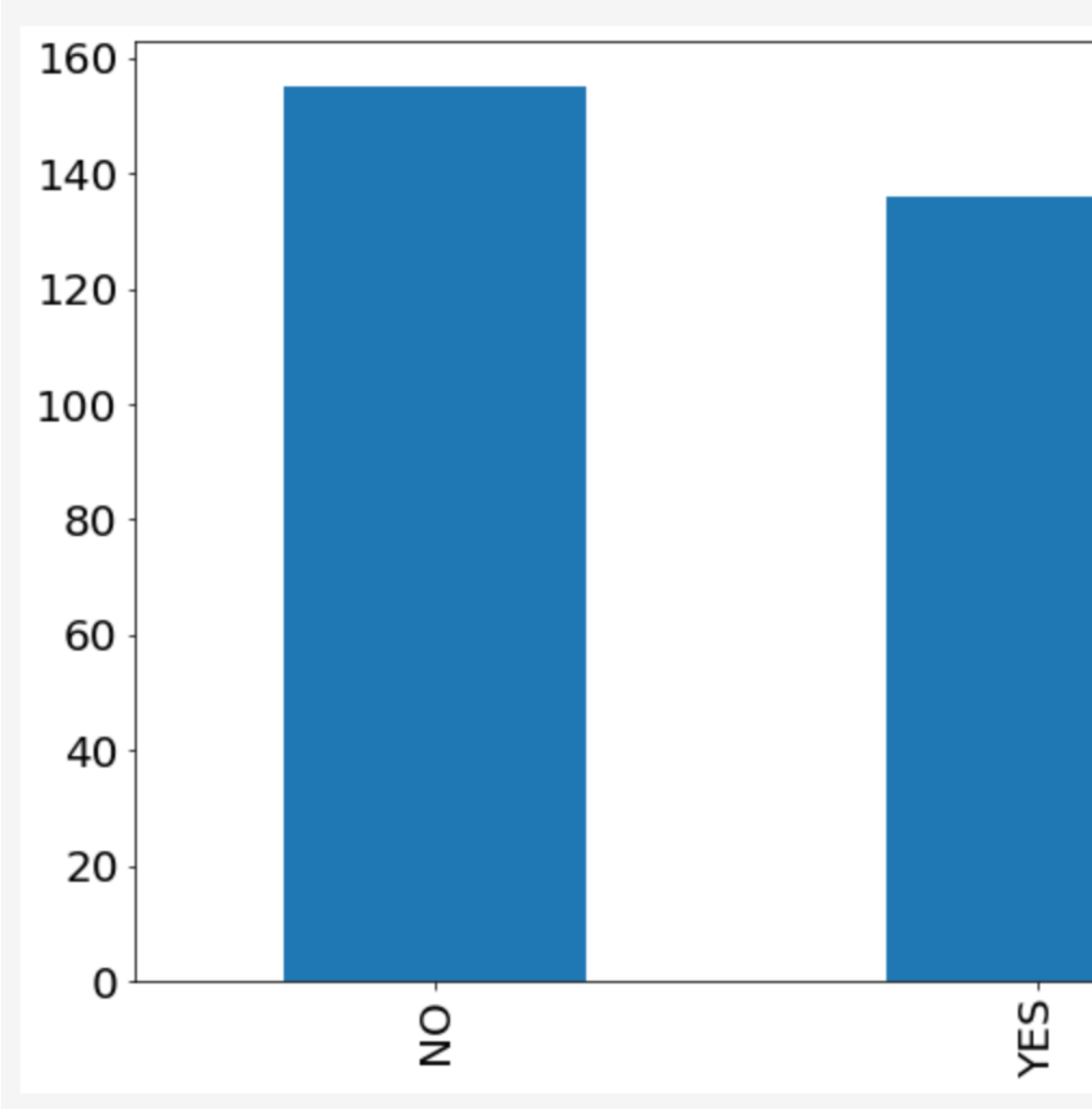
1)

a.

```
df['Depression'].value_counts()
```

✓ 0.3s

```
NO     155
YES    136
Name: Depression, dtype: int64
```



From the graph, we can know that the data is balanced data

b.

In order to detect as many patients as possible because we hope they can be treated on time. We use **recall** to identify all patients samples as much as possible. Because depression can be easily caused and treated by the environment, we regard correctness as less important and this is why we do not choose **precision.**

c.

k-fold cross validation is suitable to be used in this machine learning experiments because the the label dataset is small. When the dataset is not enough, in order to make full use of the data set, the data set a is randomly divided into k subsets, of which one subset is used as the test set for validation and the remaining k-1 subsets are used as the training set for training each iteration.

2)

create a random baseline classifier

```
dummy_uniform = DummyClassifier(strategy='uniform')

#use 5 fold cross validation to perform training and validation
#parameter scoring = 'accuracy' will compute accuracy
scores_dummy = cross_val_score(dummy_uniform, x_train, y_train,
```

3 classification algorithms

DecisionTreeClassifier, KNeighborsClassifier, SVM

Tweak the parameters

```
#Create a decision tree classifier with default parameters
dtree = DecisionTreeClassifier()

#the param_grid specifies one grid should be explored
param_grid = {"max_depth":[4, 5, 6, 7, 8]}

#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(dtree, param_grid, cv = 5)
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

```
knn = KNeighborsClassifier()

#the param_grid specifies one grid should be explored
param_grid = {"n_neighbors":[1, 3, 5, 7, 9, 11, 13, 15]}

#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(knn, param_grid, cv = 5)
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

```
#the param_grid specifies one grid should be explored
param_grid = {"C":[1, 10, 100, 1000]}

#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(SVC(kernel = 'linear'), param_grid, cv =
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

```
#the param_grid specifies one grid should be explored
param_grid = {"C":[1, 10, 100, 1000], 'degree':[1,2,3,4,5]}

#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(SVC(kernel = 'poly'), param_grid, cv = 5)
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

```
#the param_grid specifies one grid should be explored
param_grid = {"C":[1, 10, 100, 1000], 'gamma':[0.1,0.2,0.3,0.4,0.5,0.
              "coef0":[0,1,10]}
#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(SVC(kernel = 'sigmoid'), param_grid, cv =
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

```
#the param_grid specifies one grid should be explored
param_grid = {"C":[1, 10, 100, 1000], 'gamma':[0.1,0.2,0.3,0.4,0.5,0.

#fit on the dataset on all parameter combinations in param_grid
#retain the best conbination
grid_search = GridSearchCV(SVC(kernel = 'rbf'), param_grid, cv = 5)
#Train model while tuning the parameters using grid search
grid_result = grid_search.fit(x_train, y_train)
#display the best parameter c value
grid_result.best_params_
```

	DecisionTreeClassifier	KNeighborsClassifier	SVM

Freq-PHO-Binary	max_depth=7	n_neighbors=11	C=10, gamma='scale', kernel='rbf'
Norm-PHO-Binary	max_depth=5	n_neighbors=1	C=10, degree=2, kernel='poly'

According to A1_Experiment_Sheet.xlsx, Norm-PHO-Binary feature representation produces the better model, the best performing model based on recall.
The reason is that an individual may experienced multiple emotions in a day and very few emotions in another day, big data will make the small data less important, but actually the data of each day are equally important, so this may cause higher error.

4)

The best performing model can not achieve very promising results, here is the two suggestions:

1. remove outlier:

```
#predict the target for the train dataset
train_predict = dtree.predict(x_train)

#compute the model accuracy on the development set: How often
print("DecisionTreeClassifier Accuracy (train):", metrics.accu
```

✓ 0.3s

```
DecisionTreeClassifier Accuracy (train): 0.7980295566502463
```

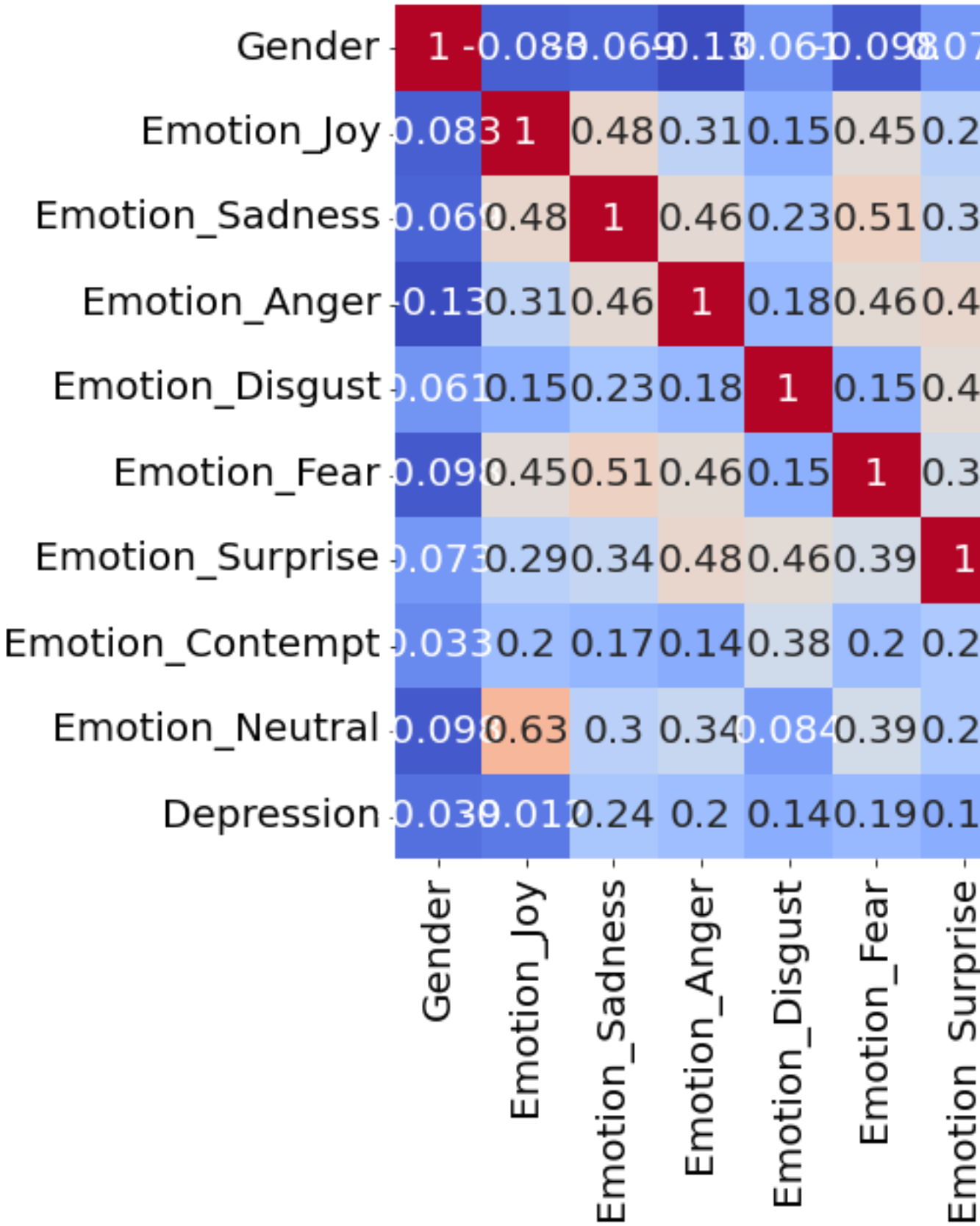
```
#predict the target for the train dataset
train_predict = knn.predict(x_train)

#compute the model accuracy on the development set: How often
print("KNeighborsClassifier Accuracy (train):", metrics.accura
```

✓ 0.9s

```
KNeighborsClassifier Accuracy (train): 0.6059113300492611
```

From the graph we know that the training accuracy is very low, so lots of outliers exists in the dataset. Outliers are unusual values, they lead to inaccurate predictions because we don't analyse the behavior and relationship with other variables correctly. Sometimes a dataset can contain extreme values that are outside the range of what is expected and unlike the other data. Often machine learning modeling and model skill in general can be improved by removing these outlier values.

2. feature creation:


Gender
Emotion_Joy
Emotion_Sadness
Emotion_Anger
Emotion_Disgust
Emotion_Fear
Emotion_Surprise
Emotion_Contempt
Emotion_Neutral
Depression
1 -0.083 -0.069 -0.13 -0.061 -0.098
0.083 1 0.48 0.31 0.15 0.45 0.2
0.069 0.48 1 0.46 0.23 0.51 0.3
0.13 0.31 0.46 1 0.18 0.46 0.4
0.061 0.15 0.23 0.18 1 0.15 0.4
0.098 0.45 0.51 0.46 0.15 1 0.3
0.073 0.29 0.34 0.48 0.46 0.39 1
0.033 0.2 0.17 0.14 0.38 0.2 0.2
0.098 0.63 0.3 0.34 0.084 0.39 0.2
0.039 0.011 0.24 0.2 0.14 0.19 0.1

feature creation helps to unleash the hidden relationship of a data set. From the graph we can know that correlation is very low, we should try to extract a higher hidden correlation to make the model better.