Precision Preprocessing and Hyperparameter Optimization & A KNN-Based Framework for College Retention Prediction

Samson Rozansky, Brian Zhang

William G. Enloe High School

IB Computer Science HL 4B

March 8th, 2024

§1 Introduction

§1.1 Research Objective

College is one of the most effective tools that facilitates class mobility, but 24% of students drop out in their first year of college. We want to reduce this amount by attempting to create a model that can detect "at-risk" students and give them the appropriate resources needed for a better future. This in turn will enhance economic opportunities for thousands of students and result in a more productive economy overall.

Most universities begin reaching out to students when they start failing courses. We believe this addresses the problem too late. We want to be very proactive at addressing these problems before their GPA start taking hits. This is because many graduate programs, internships, REU's, and many more consider GPA as a critical factor for deciding whether they should admit one student over another. So being proactive is one of the main reasons we began our inquiry.

§1.2 Research Questions

- What is the best distance to use to determine students that are "at-risk"?
- What is the most optimal "K" amount for classifying students as "at-risk"
- What are some attributes we can throw away to improve accuracy?
- What are some attributes that are important to detecting "at-risk" students?

§2 Data

Our data came from the Capacitação da Administração Pública, or Public Administration Testing in Portugal. It came with 37 different attributes, as shown below:

Feature	Data Type and Description
Curricular units 1st sem (grade)	Integer - Grade average in the 1st semester (0 to 20).

Curricular units 1st sem (without evaluations)	Integer - Number of units without evaluations in 1st semester.
Curricular units 2nd sem (credited)	Integer - Number of credited units in 2nd semester.
Curricular units 2nd sem (enrolled)	Integer - Number of units enrolled in 2nd semester.
Curricular units 2nd sem (evaluations)	Integer - Number of evaluations in 2nd semester.
Curricular units 2nd sem (approved)	Integer - Number of units approved in 2nd semester.
Curricular units 2nd sem (grade)	Integer - Grade average in 2nd semester (0 to 20).
Curricular units 2nd sem (without evaluations)	Integer - Number of units without evaluations in 2nd semester.
Unemployment rate	Continuous - Unemployment rate (%).
Inflation rate	Continuous - Inflation rate (%).
GDP	Continuous - Gross Domestic Product.
Target	Categorical - Three-category classification (dropout, enrolled, graduate).
Marital Status	Integer - Marital Status (1 – single, 2 – married, 3 – widower, etc.).
Application mode	Integer - Various application mode codes.
Application order	Integer - Application order (0 - first choice; 9 - last choice).
Course	Integer - Various course codes.
Daytime/evening attendance	Integer - 1 – daytime, 0 - evening.
Previous qualification	Integer - Education Level and related qualifications.
Previous qualification (grade)	Continuous - Grade of previous qualification (0 to 200).
Nationality	Integer - Nationality codes.
Mother's qualification	Integer - Education Level of mother.

Father's qualification	Integer - Education Level of father.
Mother's occupation	Integer - Occupation of mother.
Father's occupation	Integer - Occupation of father.
Admission grade	Continuous - Admission grade (0 to 200).
Displaced	Integer - 1 – yes, 0 – no.
Educational special needs	Integer - 1 – yes, 0 – no.
Debtor	Integer - 1 – yes, 0 – no.
Tuition fees up to date	Integer - 1 – yes, 0 – no.
Gender	Integer - Gender (1 – male, 0 – female).
Scholarship holder	Integer - 1 – yes, 0 – no.
Age at enrollment	Integer - Age of student at enrollment.
International	Integer - 1 – yes, 0 – no.
Curricular units 1st sem (credited)	Integer - Number of units credited in 1st semester.
Curricular units 1st sem (enrolled)	Integer - Number of units enrolled in 1st semester.
Curricular units 1st sem (evaluations)	Integer - Number of evaluations in 1st semester.
Curricular units 1st sem (approved)	Integer - Number of units approved in 1st semester.

Table 1: List of features, data types, and descriptions.

The data refer to records of students enrolled between the academic years 2008/2009 to 2018/2019. These include data from 17 undergraduate degrees such as design, education, nursing, journalism, management, social service, and technologies.

Shown below is an image of the correlation heatmap of the dataset, where a brighter (more yellow) color indicates stronger positive correlation and a darker (more dark purple) color indicates stronger negative correlation.

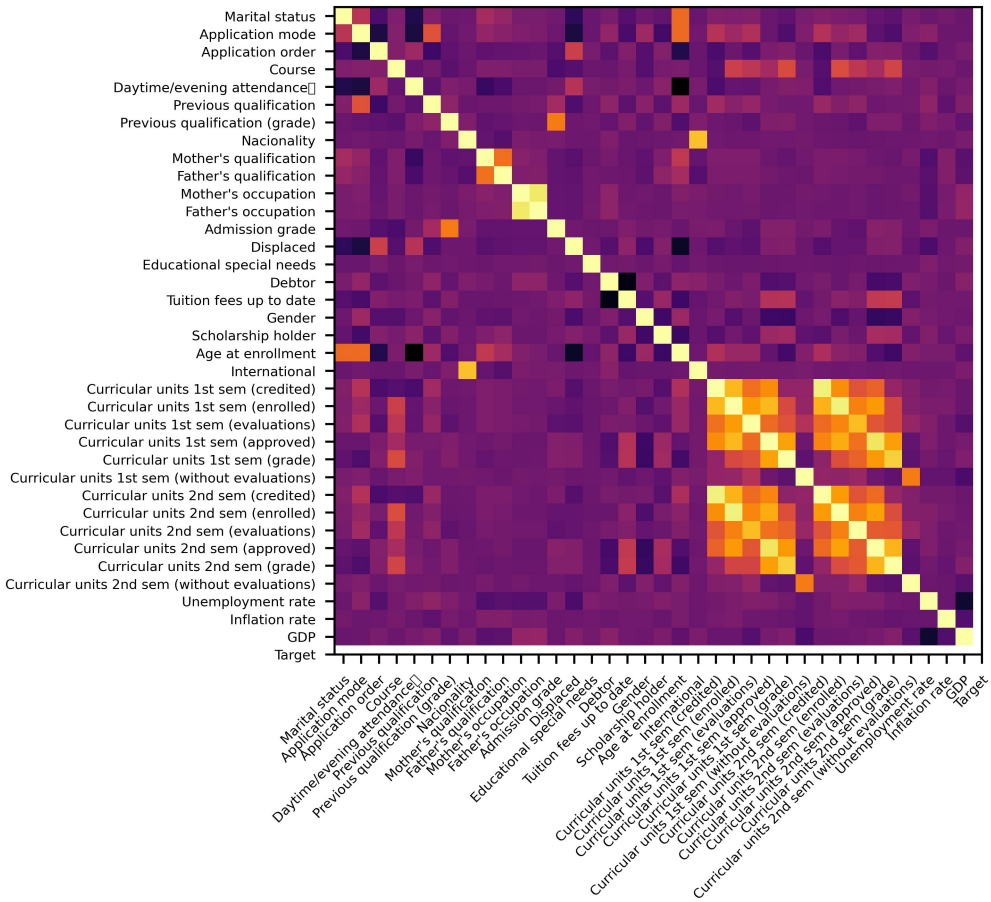

Figure 1: Correlation heatmap for the dataset. Brighter colors indicate stronger positive correlation. Dimmer colors indicate stronger negative correlation.

§3 Preprocessing

Before the data in the dataset can be used in models, we must prepare it for analysis by cleaning, transforming, and formatting it in a way that makes it suitable for the models. Raw data often is incomplete, has errors, and is inconsistent, and this can lead to biased models or incompatability with certain models. Data preprocessing is the process of dealing with the issues with raw data, and preparing it for use in models. This ensures that machine learning models can be used and are reliable.

The first error we ran into, was that the data used semi-colons as delimiters, when we need commas, since CSV stands for Comma Seperated Values.

The second error we ran into, was that the data contained students who were currently enrolled. This is not useful for the exploration since we are trying to predict whether a student will drop out or not, and currently enrolled students would make no sense as a target value

The last error we ran into, was that the data varied a lot, and would benefit if it were normalized

§3.1 Normalization

There are large variations in the magnitude of data, with magnitudes of data ranging from nearly 10,000 to less than 1. Shown below is a box plot of the data: This difference in magnitude of data can confound the distance metrics, as the model may believe that the features with magnitude 10,000 are far more important than the features with magnitude 1.

In order to deal with this, the data was normalized based on the z-scores of each data point in each feature. For a given feature, let μ be the average of all data points of the feature and σ be the standard deviation of all data points of the feature. Then, the formula for the z-score of some arbitrary data point x in the feature is as follows:

$$z = \frac{x - \mu}{\sigma}.$$

Each data point was then converted into its z-score and a new dataset was created, with the following box plot. This yields a normalized range of values for the model

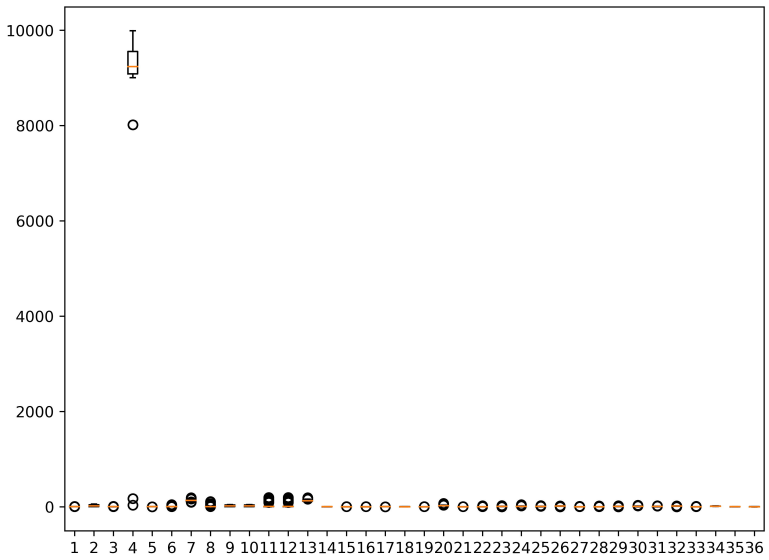

Figure 2: Boxplot of raw data prior to normalization.

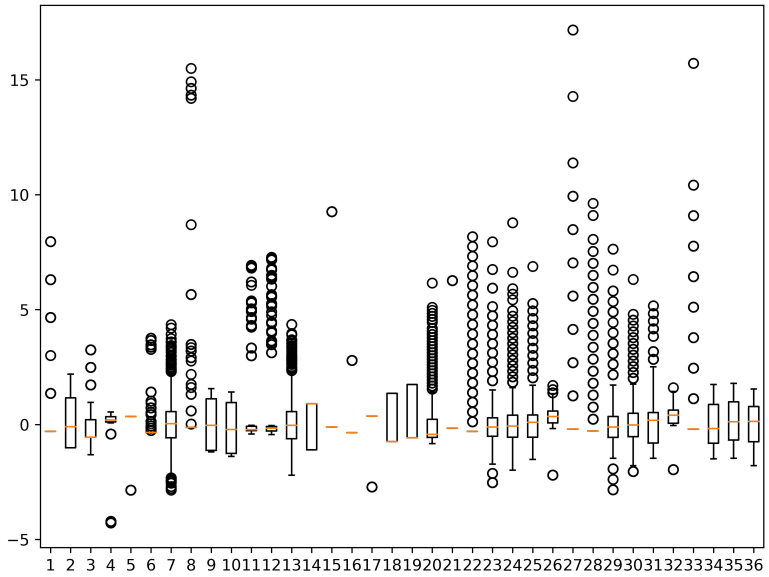

Figure 3: Boxplot of data after z-score normalization.

to be trained on. In addition to this, the normalized dataset has the benefit of each feature having mean $\mu = 0$ and standard deviation $\sigma = 1$.

§3.2 Recursive Feature Elimination

Recursive feature elimination (RFE) is a method used to find the optimal set of features within a dataset. This method works recursively by following these steps:

1. In the dataset with n features, the model is trained on each distinct group of $n - 1$ features.
2. The accuracy for each of the groups of $n - 1$ features is recorded.
3. The group of $n - 1$ features with the highest accuracy is kept and becomes the new dataset.
4. Steps 1-3 are repeated until there is 1 feature left in the dataset.
5. The set of features with the highest accuracy out of all the tests is determined.

§3.2.1 Benefits

Some benefits of this are that we can see which features we can ignore and remove them to improve overall accuracy all while using a relatively simple method and algorithm. These features may be confounding the model by providing unnecessary information.

The lower amount of features also means that there is less data for the model to be trained on, thereby decreasing training time and processing resources required.

§3.2.2 Limitations

Some limitations of this are that we may not be able to create the most accurate group of features using RFE. Removing two features with smaller accuracy increases might result in greater accuracy gains than removing the one feature that provided the greatest accuracy increase.

In order to test every possible group of features that can be used, the time complexity is of magnitude $O(2^N)$, which is far too time-consuming for feasible testing.

Name	Formula
L1	$\sum \lvert x_i - y_i \rvert$
L2	$\sqrt{\sum (x_i - y_i)^2}$
Cosine	$1 - \cos\theta = 1 - \frac{X \cdot Y}{\lvert\lvert X \rvert\rvert \lvert\lvert Y \rvert\rvert}$

Table 2: Distance metrics and formulae

Name	k	Accuracy
Cosine	16	0.8705
L1	10	0.8788
L2	6	0.8664

Table 3: Best values of k for each of the models

§4 Model Training

We used three different distance metrics to compare the entries in the model.

For each of the distances, we tested k from 1 to 20, and determined which k had the greatest accuracy. The optimal values are below: Shown below is a graph of the k value and the accuracy of the model for predicting the training dataset and the testing dataset.

§5 Results

The following figures present the results of our models: Out of the three distance

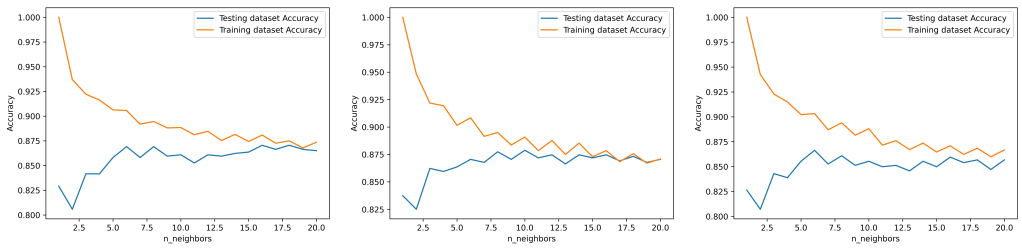

Figure 4: Graph of the k value and accuracy for the three models (Cosine, L1, L2 from left to right)

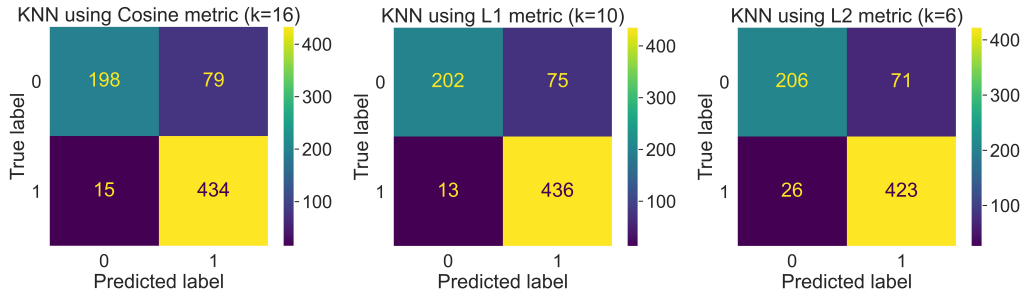

Figure 5: The confusion Matrices for Cosine, L1, and L2 distance metrics

Model	Accuracy	Recall	Precision
Cosine	0.8705	0.9666	0.8460
L1	0.8788	0.9710	0.8532
L2	0.8664	0.9421	0.8563

Table 4: Various model metrics compared to distance metric

metrics, L1 had the greatest accuracy with 87.88%. Shown below is a detailed breakdown of the metrics of the model.

The ROC metric is to check the fit of your model. It is a plot of the True Positive Rate against the False Positive Rate. The model performance is determined by looking at the area under the ROC curve. The model with the best ROC is L1.

Afterwards, RFE was used to determine the optimal set of features. Results are shown:

Metric	Accuracy	Number of features
Cosine	0.9045	12
L1	0.9105	14
L2	0.9105	13

Table 5: Detailed results after RFE

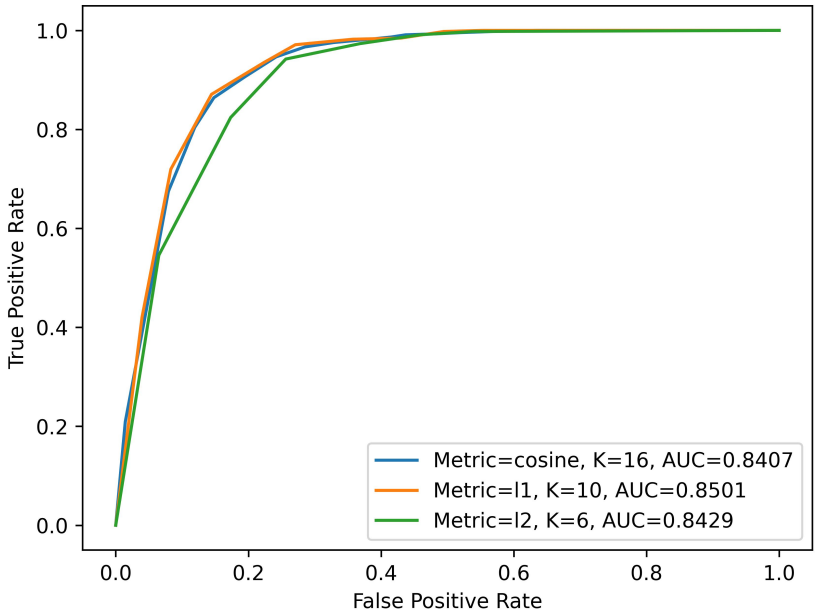

Figure 6: ROC curve with AUC

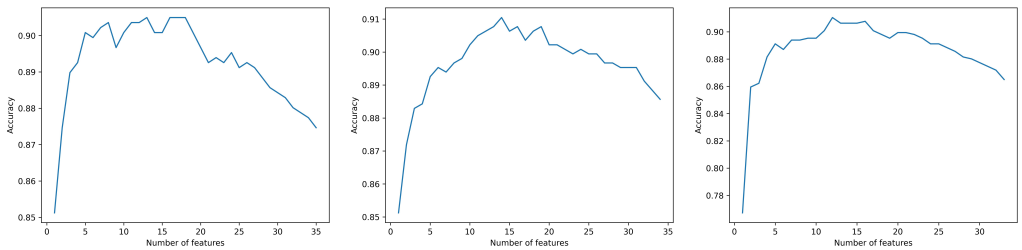

Figure 7: Graphs of Recursive Feature Elimination with the three different distance metrics

§6 Conclusion

Predicting whether people will drop out or not through our KNN-based model and preprocessing techniques can have a significant positive impact. By identifying at-risk students early, universities can provide targeted support before issues escalate to dropouts. This proactive approach increases student success rates and unlocks greater economic opportunities. Ultimately, this data-driven research highlights the potential of predictive analytics in building more equitable and supportive higher education systems.

In future studies about this topic, instead of sourcing it only from the Portuguese "Capacitação da Administração", we want to source the information from around the world including countries that are in different geological regions, cultural norms, and much more. This would let us make sure we aren't being influenced by these factors, and we could see which factors influence whether a student will drop out or not.

§7 Works Cited and Links

Hanson, Melanie. "College Dropout Rate [2023]: by Year + Demographics." Education Data Initiative, 29 October 2023, https://educationdata.org/college-dropout-rates. Accessed 11 March 2024.

"Predict Students' Dropout and Academic Success." UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success. Accessed 11 March 2024.

Rose, Marley. "College Dropout Rate in the U.S." BestColleges.com, https://www.bestcolleges.com/research/college-dropout-rate/. Accessed 11 March 2024.

The GitHub repo https://github.com/samson-rozansky/IB-Computer-Science-KNN

§A Code

§A.1 Preprocessing

```
import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("data")

data = pd.read_csv(ROOT.joinpath("data_raw.csv"))

data_normal = data.copy()
for column in data_normal.columns[:36]:
    data_normal[column] = (data_normal[column] - data_normal[column].mean())/data_normal[column].std(ddof=0)

ret = []
for index, row in data_normal.iterrows():
    if (row["Target"] != "Enrolled"):
        if (row["Target"] == "Graduate"):
            row["Target"] = 1
        else:
            row["Target"] = 0
        ret.append(row)

df = pd.DataFrame(ret)
df.reset_index()
df.to_csv(ROOT.joinpath("data_normal.csv"), index = False)
```

§A.2 Boxplot Raw

```
import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd

```

```
ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("figs")

data = pd.read_csv(ROOT.joinpath("data_raw.csv"))

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(data.iloc[:, :36])
plt.savefig(OUTPUT.joinpath("boxplot_raw.jpg"), bbox_inches = "tight", transparent = True, dpi = 600)
```

§A.3 Boxplot Normalized

```
import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("figs")

DATA_FILE = ROOT.joinpath("data_normal.csv")

if not DATA_FILE.is_file():
    import preprocess

df = pd.read_csv(DATA_FILE)

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(df.iloc[:, :36])
plt.savefig(OUTPUT.joinpath("boxplot_normal.jpg"), bbox_inches = "tight", transparent = True, dpi = 600)
```

§A.4 Correlation Heatmap

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pathlib

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("figs")

DATA_FILE = ROOT.joinpath("data_raw.csv")

df = pd.read_csv(DATA_FILE)

correlations = np.round(df.corr().to_numpy(), decimals=2)

#plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')
#plt.style.use(['dark_background'])

plt.rcParams.update({'font.size': 5})

fig, ax = plt.subplots()
im = ax.imshow(correlations, cmap = 'inferno')

# Show all ticks and label them with the respective list entries
ax.set_xticks(np.arange(len(list(df.columns))), labels=list(df.columns))
ax.set_yticks(np.arange(len(list(df.columns))), labels=list(df.columns))

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
    rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
"""
for i in range(len(list(df.columns))):
```

```
    for j in range(len(list(df.columns))):
        text = ax.text(j, i, correlations[i, j], ha="center", va="center", color="aqua")
"""

fig.tight_layout()

plt.savefig(OUTPUT.joinpath("heatmap.jpg"), bbox_inches = "tight", transparent = True, dpi = 600)
```

§A.5 Hyperparameter Optimization

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

import os
import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import precision_score, accuracy_score, confusion_matrix, ConfusionMatrixDisplay

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("figs\distance")

DATA_FILE = ROOT.joinpath("data_normal.csv")

if not DATA_FILE.is_file():
    import preprocess

data = pd.read_csv(DATA_FILE)

# Create feature and target arrays
```

```
X = data.drop(columns = ['Target'])
y = data['Target']

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state=42)

metrics_list = ['cosine', 'l1', 'l2']
name = ['Cosine', 'L1', 'L2']

for m in range(3):
    neighbors = np.arange(1, 21)
    train_accuracy = np.empty(len(neighbors))
    test_accuracy = np.empty(len(neighbors))
    best_acc = 0
    best_k = 0
    for i, k in enumerate(neighbors):
        model = KNeighborsClassifier(n_neighbors=k, metric=
    metrics_list[m])
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        cur_acc = metrics.accuracy_score(y_test, predictions)
        if (cur_acc > best_acc):
            best_acc = cur_acc
            best_k = k

        train_accuracy[i] = model.score(X_train, y_train)
        test_accuracy[i] = model.score(X_test, y_test)

    print('Best k for ' + name[m] + ': ' + str(best_k))

    model = KNeighborsClassifier(n_neighbors=best_k, metric=
    metrics_list[m])
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    write_file = OUTPUT.joinpath(name[m] + "_result.txt")
    if not write_file.is_file():
```

```
60          open(write_file, 'x')
61      with open(write_file, 'w') as f:
62          f.write('Evaluation Metrics for ' + name[m] + ':\n')
63          f.write('Accuracy: ' + str(metrics.accuracy_score(y_test,
    predictions)) + '\n')
64          f.write('Recall: ' + str(metrics.recall_score(y_test,
    predictions)) + '\n')
65          f.write('F1 Score: ' + str(metrics.f1_score(y_test,
    predictions)) + '\n')
66          f.write('Precision: ' + str(metrics.precision_score(y_test
    , predictions)) + '\n')
67
68      # Confusion Matrix
69      cm = confusion_matrix(y_test, predictions, labels = None)
70      #sns.set(font_scale=2)
71      disp = ConfusionMatrixDisplay(confusion_matrix = cm,
    display_labels = None)
72      disp.plot()
73      plt.grid(False)
74      plt.title("KNN using " + name[m] + " metric (k=" + str(best_k)
        + ")")
75      #plt.savefig(OUTPUT.joinpath(name[m] + "_confusion_matrix.jpg
    "), bbox_inches = "tight", transparent = True, dpi = 600)
76      plt.close()
77
78      # graph of k vs accuracy
79      plt.plot(neighbors, test_accuracy, label = name[m] + ' Testing
        dataset Accuracy')
80      plt.plot(neighbors, train_accuracy, label = name[m] + '
    Training dataset Accuracy')
81
82      plt.legend()
83      plt.xlabel('n_neighbors')
84      plt.ylabel('Accuracy')
85      #plt.savefig(OUTPUT.joinpath(m + "_neighbors_vs_accuracy.jpg")
    , bbox_inches = "tight", transparent = True, dpi = 600)
86      #plt.close()
87  plt.savefig(OUTPUT.joinpath("neighbors_vs_accuracy.jpg"),
```

```
    bbox_inches = "tight", transparent = True, dpi = 600)
```

§A.6 ROC Curve

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import precision_score, accuracy_score,
    confusion_matrix, ConfusionMatrixDisplay

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("
    data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("
    figs")

DATA_FILE = ROOT.joinpath("data_normal.csv")

if not DATA_FILE.is_file():
    import preprocess

data = pd.read_csv(DATA_FILE)

# Create feature and target arrays
X = data.drop(columns = ['Target'])
y = data['Target']

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state=42)

metrics_list = ['cosine', 'l1', 'l2']
k = [16, 10, 6]
```

```

for i in range(3):
    model = KNeighborsClassifier(n_neighbors=k[i], metric=
    metrics_list[i])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[::, 1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = round(metrics.roc_auc_score(y_test, y_pred), 4)

    plt.plot(fpr, tpr, label = "Metric=" + metrics_list[i] + ", K=
    " + str(k[i]) + ", AUC=" + str(auc))

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.savefig(OUTPUT.joinpath("roc_curve.jpg"), bbox_inches = "tight
    ", transparent = True, dpi = 600)
```

§A.7 Recursive Feature Elimination

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
import pathlib
import pandas as pd
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import precision_score, accuracy_score,
    confusion_matrix, ConfusionMatrixDisplay

ROOT = pathlib.Path(__file__).parent.parent.resolve().joinpath("
    data")
OUTPUT = pathlib.Path(__file__).parent.parent.resolve().joinpath("
    figs\\rfe")

```

```
DATA_FILE = ROOT.joinpath("data_normal.csv")

if not DATA_FILE.is_file():
    import preprocess

data = pd.read_csv(DATA_FILE)

features = []
accuracies = []

def RFE(df, m, k):
    best_acc = 0
    feature = 'temp'
    for column in df:
        if (column == 'Target'):
            continue
        temp = df.copy()
        temp.drop(columns = [column], inplace = True)
        X = temp.drop(columns = ['Target'])
        y = temp['Target']

        X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state=42)
        model = KNeighborsClassifier(n_neighbors=k, metric=m)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        cur_acc = metrics.accuracy_score(y_test, y_pred)
        if (cur_acc > best_acc):
            feature = column
            best_acc = cur_acc

    df.drop(columns = [feature], inplace = True)
    features.append(feature)
    accuracies.append(best_acc)

    if (len(df.columns) > 2):
        print("Number eliminated: " + str(len(features)))
        print("Number remaining: " + str(len(df.columns)) + "\n")
```

```
            RFE(df.copy(), m, k)
            return
        else:
            return

metrics_list = ['cosine', 'l1', 'l2']
k = [16, 10, 6]

for i in range(3):
    RFE(data, metrics_list[i], k[i])
    write_file = OUTPUT.joinpath(metrics_list[i] + "_RFE.txt")
    if not write_file.is_file():
        open(write_file, 'x')
    open(write_file, 'w').close()

    for j in range(len(features)):
        with open(write_file, 'a') as f:
            f.write("Removed: " + features[j] + "\nAccuracy: " +
    str(accuracies[j]) + "\n\n")

    plt.plot(np.arange(len(features), 0, -1), accuracies)
    plt.xlabel('Number of features')
    plt.ylabel('Accuracy')

    plt.savefig(OUTPUT.joinpath(metrics_list[i] + "_RFE.jpg"),
    bbox_inches = "tight", transparent = True, dpi = 600)
    plt.close()

    features = []
    accuracies = []
```