MET CS677 Data Science with Python
Assignment 6

Alan Szmyt

April 24, 2023

In this assignment, you will implement k-means clustering and use it to construct a multi-label classifier to determine the variety of wheat.

For the dataset, we use "seeds" dataset from the machine learning depository at UCI:

https://archive.ics.uci.edu/ml/datasets/seeds.

Dataset Description: From the website: "...The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment..."

There are 7 (continuous) features: $F = \{f_1, \ldots, f_7\}$ and a class label L (Kama: 1, Rosa: 2, Canadian: 3).

1. f_1: area A
2. f_2: perimeter P
3. f_3: compactness $C = 4\pi A/P^2$
4. f_4: length of kernel
5. f_5: width of kernel
6. f_6: asymmetry coefficient
7. f_7: length of kernel groove
8. L: class (Kama: 1, Rosa: 2, Canadian: 3)

For the first question, you will choose 2 class labels as follows. Take the last digit in your BUID and divide it by 3. Choose the following 2 classes depending on the remainder R:

1. $R = 0$: class $L = 1$ (negative) and $L = 2$ (positive).
2. $R = 1$: class $L = 2$ (negative) and $L = 3$ (positive).
3. $R = 2$: class $L = 1$ (negative) and $L = 3$ (positive).

1 **Question 1:** Take the subset of the dataset containing your two class labels. You will use random $50/50$ splits for training and testing data.

Cell 03

```
import operator
import pandas as pd
import seaborn as sns
from pathlib import Path
from pandas import DataFrame
from IPython.display import Latex
from assignment6 import (
    log, create_latex_table, data_directory, LabelClasses, SeedsColumn,
    DATAFRAME_NAME, SEEDS_DATASET, SEEDS_SUBSET, LinearSVMModel,
    PolynomialSVMModel, GaussianSVMModel, KNNModel,
    AnalyticsCollection, ModelAnalyzer, ClassifierAnalytics,
    KMeansAnalyticsCollection, KMeansAnalytics, KMeansModel,
    WheatClass, KMEANS_CLASSIFIER_K,
)

# Global Seaborn options.
sns.set_theme(font_scale=1.5, rc={"text.usetex": True})
pd.set_option("display.precision", 2)
pd.set_option("styler.format.precision", 2)

# Choosing the 2 classes based upon BUID.
buid: str = "U38573068"
label_classes: LabelClasses = LabelClasses.select_classes(
    remainder=int(int(buid[-1]) % 3)
)

# seeds dataset file from UCI.
dataset_csv: str = "seeds_dataset.csv"
seeds_dataset_file: Path = data_directory.joinpath(dataset_csv)
```

Cell 04

```
# Load the whole dataset from the csv file.
seeds_dataset: DataFrame = pd.read_csv(
    filepath_or_buffer=seeds_dataset_file,
    delimiter="\t",
    engine="python",
    header=None,
    names=SeedsColumn.columns(),
    dtype=SeedsColumn.dtype()
)
seeds_dataset.attrs[DATAFRAME_NAME] = SEEDS_DATASET
```

Cell 05

```
# Create subset of dataset using the 2 selected classes.
seeds_subset: DataFrame = \
    seeds_dataset[seeds_dataset[SeedsColumn.CLASS].isin(label_classes.classes)]
seeds_subset.attrs[DATAFRAME_NAME] = SEEDS_SUBSET

# Machine learning model analyzer context class.
analyzer: ModelAnalyzer = ModelAnalyzer(
    dataset=seeds_subset, model=LinearSVMModel(predictor_col=SeedsColumn.CLASS)
)
```

Cell 06

```
seeds_subset_table: str = create_latex_table(
    seeds_subset.head(),
    label="tab:seeds_subset_datatable",
    caption="Seeds Subset"
)
Latex(seeds_subset_table)
```

Table 1: Seeds Subset

area	perimeter	compactness	length	width	asymmetry	groove	class
15.26	14.84	0.87	5.76	3.31	2.22	5.22	1
14.88	14.57	0.88	5.55	3.33	1.02	4.96	1
14.29	14.09	0.91	5.29	3.34	2.70	4.83	1
13.84	13.94	0.90	5.32	3.38	2.26	4.80	1
16.14	14.99	0.90	5.66	3.56	1.35	5.17	1

1.1 Implement a linear kernel SVM. What is your accuracy and confusion matrix?

Answer:

Cell 07

```
# Train linear svm model, make predictions, and gather analytics.
linear_svm_analytics: ClassifierAnalytics = analyzer.analyze()
```

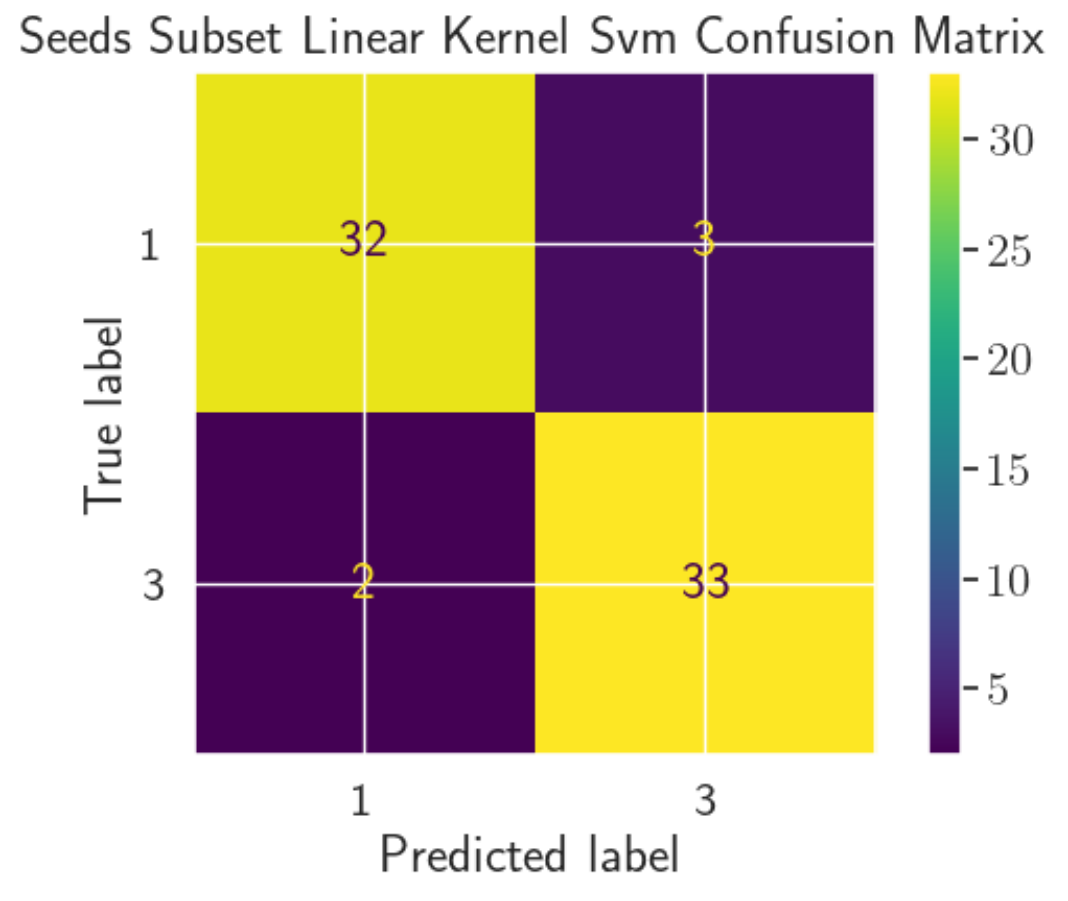
Cell 08

```
    # Compute the accuracy.
log(
    f"Linear kernel SVM accuracy: "
    f"{linear_svm_analytics.confusion_matrix.accuracy.score}"
)
```

```
Linear kernel SVM accuracy: 92.86%
```

Cell 09

```
# Compute the confusion matrix.
linear_svm_analytics.show_confusion_matrix()
```



1.2 Implement a Gaussian kernel SVM. What is your accuracy and confusion matrix?

Answer:

Cell 10

```
# Train gaussian svm model, make predictions, and gather analytics.
analyzer.model = GaussianSVMModel(predictor_col=SeedsColumn.CLASS)
gaussian_svm_analytics: ClassifierAnalytics = analyzer.analyze()
```

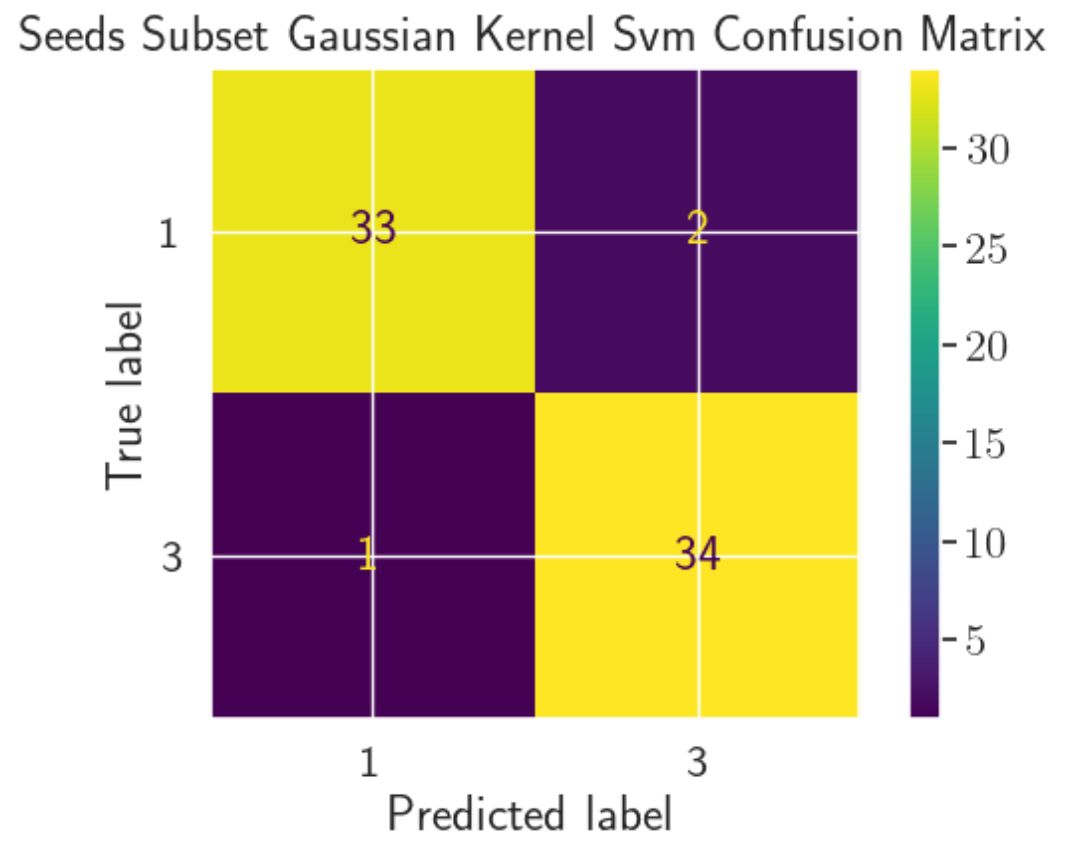
Cell 11

```
# Compute the accuracy.
log(
    f"Gaussian kernel SVM accuracy: "
    f"{gaussian_svm_analytics.confusion_matrix.accuracy.score}"
)
```

```
Gaussian kernel SVM accuracy: 95.71%
```

Cell 12

```
# Compute the confusion matrix.
gaussian_svm_analytics.show_confusion_matrix()
```



1.3 Implement a polynomial kernel SVM of degree 3. What is your accuracy and confusion matrix?

Answer:

Cell 13

```
# Train polynomial svm model, make predictions, and gather analytics.
analyzer.model = PolynomialSVMModel(predictor_col=SeedsColumn.CLASS)
polynomial_svm_analytics: ClassifierAnalytics = analyzer.analyze()
```

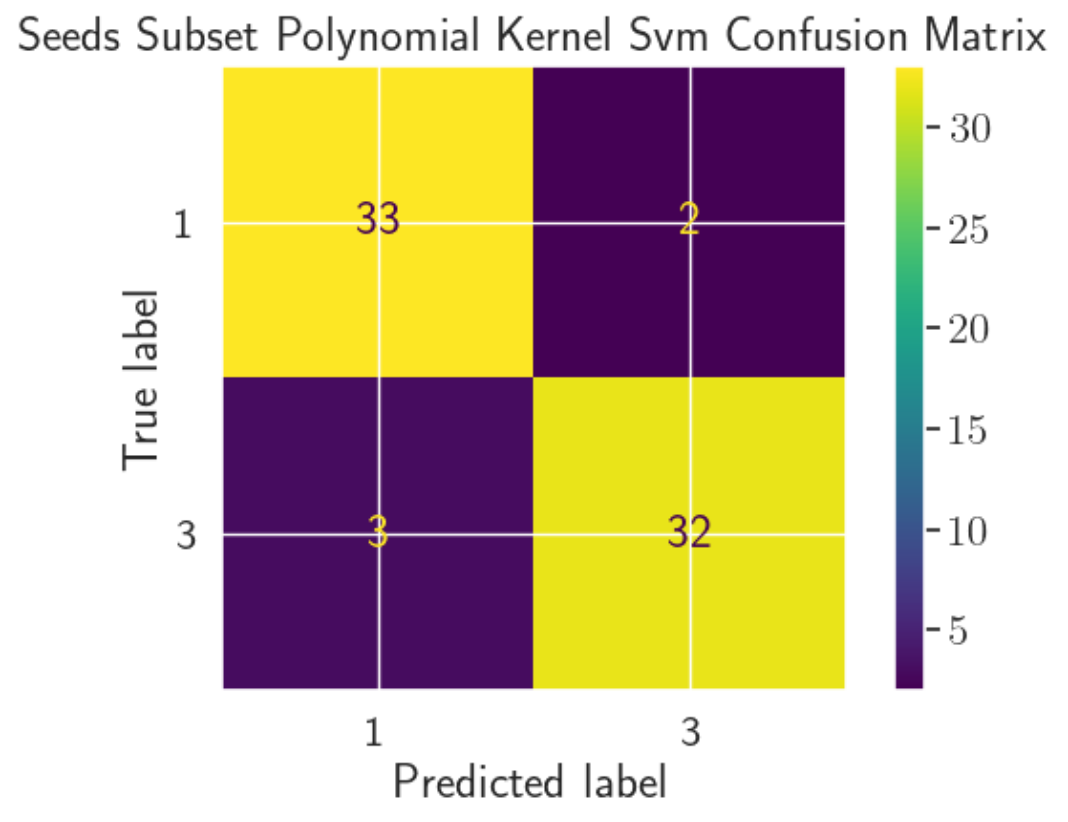
Cell 14

```
# Compute the accuracy.
log(
    f"Polynomial kernel SVM accuracy: "
    f"{polynomial_svm_analytics.confusion_matrix.accuracy.score}"
)
```

```
Polynomial kernel SVM accuracy: 92.86%
```

Cell 15

```
# Compute the confusion matrix.
polynomial_svm_analytics.show_confusion_matrix()
```



2 Question 2: Pick up any classifier for supervised learning (e.g. kNN, logistic regression, Naive Bayesian, etc.)

2.1 Use this classifier to your dataset. What is your accuracy and confusion matrix?

Answer:

Cell 16

```
# Train kNN model, make predictions, and gather analytics.
analyzer.model = KNNModel(predictor_col=SeedsColumn.CLASS)
knn_analytics: ClassifierAnalytics = analyzer.analyze()
```

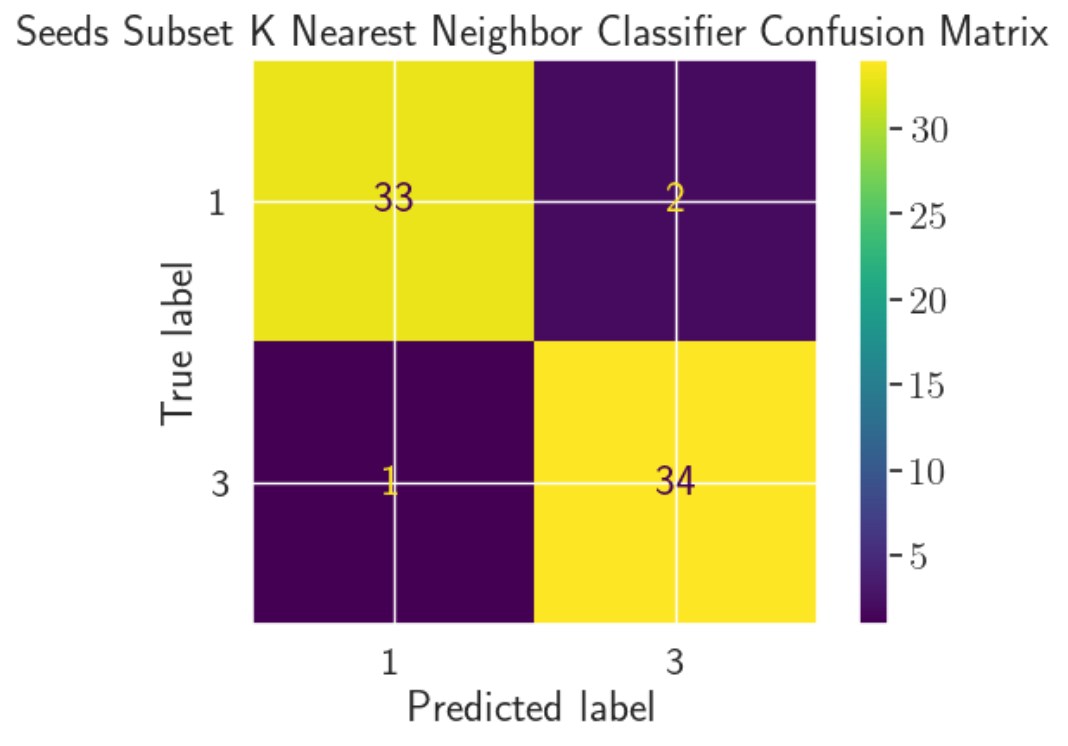
Cell 17

```
# Compute the accuracy.
log(
    f"kNN classifier accuracy: "
    f"{knn_analytics.confusion_matrix.accuracy.score}"
)
```

```
kNN classifier accuracy: 95.71%
```

Cell 18

```
# Compute the confusion matrix.
knn_analytics.show_confusion_matrix()
```



2.2 Summarize your findings in a table below and discuss your results.

Answer:

Cell 19

```
# Gather analytics for all classifier models.
classifier_analytics: AnalyticsCollection = AnalyticsCollection(
    [
        linear_svm_analytics,
        gaussian_svm_analytics,
        polynomial_svm_analytics,
        knn_analytics,
    ]
)
```

Cell 20

```
# Generate the summary table from the analytics.
summary_table_df: DataFrame = classifier_analytics.summary_table
summary_table_df.attrs[DATAFRAME_NAME] = "classifier_summary_table"
summary_table: str = create_latex_table(
    summary_table_df.head(),
    label="tab:summary table",
    caption="Classifier Statistics Summary"
)
Latex(summary_table)
```

Table 2: Classifier Statistics Summary

Model	TP	FP	TN	FN	accuracy	TPR	TNR
Linear Kernel SVM	33	3	32	2	0.93	0.94	0.91
Gaussian Kernel SVM	34	2	33	1	0.96	0.97	0.94
Polynomial Kernel SVM	32	2	33	3	0.93	0.91	0.94
k-Nearest Neighbor Classifier	34	2	33	1	0.96	0.97	0.94

3 Question 3: Take the original dataset with all 3 class labels.

3.1 For $k = 1, 2, \ldots, 8$ use k-means clustering with random initialization and defaults. Compute and plot distortion vs. k. Use the "knee" method to find the best k.

Answer:

Cell 21

```
# Train k-means clustering model, make predictions, and gather analytics.
analyzer.dataset = seeds_dataset
k_means_list: list[KMeansAnalytics] = []
for k in range(1, 8):
    analyzer.model = KMeansModel(
        predictor_col=SeedsColumn.CLASS,
        n_clusters=k,
        features=SeedsColumn.feature_columns(),
    )
    k_means_list.append(analyzer.analyze())

k_means_analytics: KMeansAnalyticsCollection = \
    KMeansAnalyticsCollection(k_means_list)
```

Cell 22

```
# Use the 'knee' method to find the best k.
optimal_k: int = k_means_analytics.show_knee_plot()
```

Cell 23

```
# The optimal k value found by the kneedle algorithm.
log(f"Optimal k value: {optimal_k}")
```

```
Optimal k value: 3
```

3.2 Re-run your clustering with best k clusters. Pick two features f_i and f_j at random (using python, of course) and plot your datapoints (different color for each class and centroids) using f_i and f_j as axis. Examine your plot. Are there any interesting patterns?

Answer:

Cell 24

```
# Get two random feature columns, f_i and f_j.
random_features: list[str] = SeedsColumn.random_columns()[:2]

log(f"Random features: {', '.join(random_features)}")
```

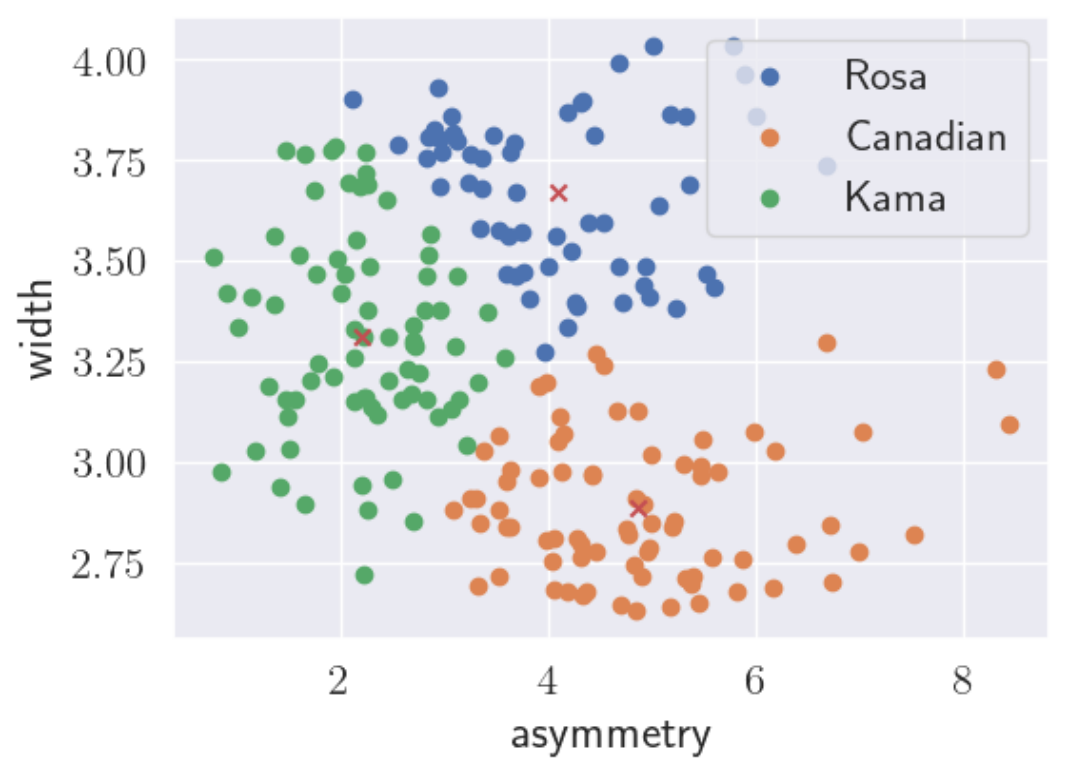
```
Random features: asymmetry, width
```

Cell 25

```
# Use optimal k value for k-means clustering.
analyzer.model = KMeansModel(
    predictor_col=SeedsColumn.CLASS,
    n_clusters=optimal_k,
    features=random_features,
    label_transformer=WheatClass.from_label
    if optimal_k == KMEANS_CLASSIFIER_K else None,
)
optimal_k_means_analytics: KMeansAnalytics = analyzer.analyze()
```

Cell 26

```
# Plot the random features.
optimal_k_means_analytics.plot(random_features, WheatClass.label_to_title)
```



After examining the plot, it seems that the Canadian seeds have small widths and are asymmetrical. The Kama seeds have more symmetry and a large range of widths. The Rose seeds have a higher width overall and are asymmetrical.

3.3 For each cluster, assign a cluster label based on the majority class of items. For example, if cluster C_i contains 45% of class 1 ("Kama" wheat), 35% of class 2 ("Rosa" wheat) and 20% of class 3 ("Canadian" wheat), then this cluster C_i is assigned label 1. For each cluster, print out its centroid and assigned label.

Answer:

Cell 27

```
# Use all features again.
analyzer.model = KMeansModel(
    predictor_col=SeedsColumn.CLASS,
    n_clusters=optimal_k,
    features=SeedsColumn.feature_columns(),
    label_transformer=WheatClass.from_label
)

# Print cluster centroid and assigned label.
for i, cluster in enumerate(optimal_k_means_analytics.clusters):
    log(
        f"Cluster {i+1}: "
        f"Centroid={cluster.centroid}, "
        f"Label={WheatClass.from_label(cluster.majority).to_title()}"
    )
```

```
Cluster 1: Centroid=[4.10076271 3.67005085], Label=Rosa
Cluster 2: Centroid=[4.87348684 2.88577632], Label=Canadian
Cluster 3: Centroid=[2.19616267 3.31273333], Label=Kama
```

3.4 Consider the following multi-label classifier. Take the largest 3 clusters with label 1, 2, and 3 respectively. Let us call these clusters A, B, and C. For each of these clusters, you know their means (centroids): $\mu(A)$, $\mu(B)$, and $\mu(C)$. We now consider the following procedure (conceptually analogous to nearest neighbor with $k = 1$): for every point x in your dataset, assign a label based on the label on the nearest (using Euclidean distance) centroid of A, B, or C. In other words, if x is closest to center of cluster A, you assign it label 1. If x is closest to center of cluster B, you assign it class 2. Finally, if x is closest to center of cluster C, you assign it class 3. What is the overall accuracy of this new classifier when applied to the complete data set?

Answer:

Cell 28

```
if optimal_k < KMEANS_CLASSIFIER_K:
    # Rerun with k=3.
    analyzer.model = KMeansModel(
        predictor_col=SeedsColumn.CLASS,
        n_clusters=3,
        features=random_features,
        label_transformer=WheatClass.from_label,
    )
elif optimal_k > KMEANS_CLASSIFIER_K:
    # Find the largest 3 clusters.
    clusters = [
        cluster.dataset
        for cluster in sorted(
            optimal_k_means_analytics.clusters,
            key=operator.attrgetter("size"),
            reverse=True,
        )[:3]
    ]

    # Run k-means with largest 3 clusters.
    analyzer.dataset = pd.concat(clusters)
```

Cell 29

```
k_means_classifier_analytics: KMeansAnalytics = analyzer.analyze()

# Compute the accuracy.
log(
    f"K-means classifier accuracy: "
    f"{k_means_classifier_analytics.confusion_matrix.accuracy.score}"
)
```

```
K-means classifier accuracy: 97.22%
```

3.5 Take this new classifier and consider the same two labels that you used for SVM. What is your accuracy and confusion matrix? How does your new classifier (from task 4) compare with any classifiers listed in the table for question 2 above?

Answer:

Cell 30

```
# Use the subset dataframe again.
analyzer.dataset = seeds_subset
k_means_classifier_subset_analytics: KMeansAnalytics = analyzer.analyze()
```

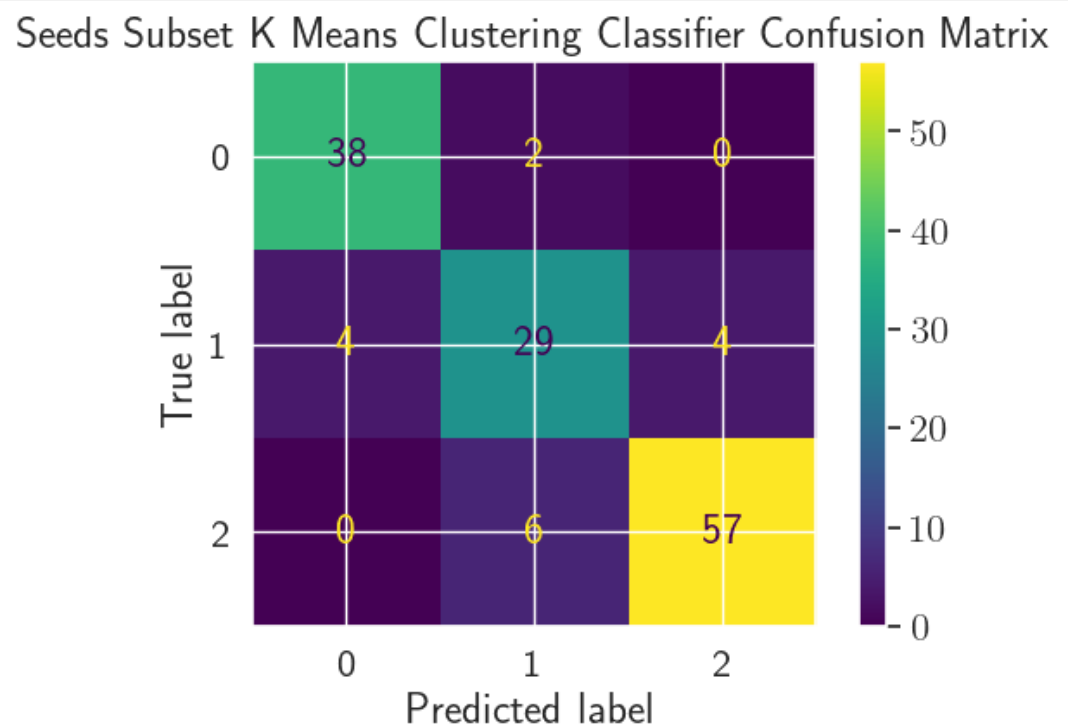
Cell 31

```
# Compute the accuracy.
log(
    f"K-means classifier on subset accuracy: "
    f"{k_means_classifier_subset_analytics.confusion_matrix.accuracy.score}"
)
```

```
K-means classifier on subset accuracy: 95.45%
```

Cell 32

```
# Compute the confusion matrix.
k_means_classifier_subset_analytics.show_confusion_matrix()
```



Cell 33

```
# Generate the summary table from the analytics.
classifier_analytics.append(k_means_classifier_subset_analytics)
final_summary_df: DataFrame = classifier_analytics.summary_table
final_summary_df.attrs[DATAFRAME_NAME] = "final_classifier_summary_table"
final_summary_table: str = create_latex_table(
    final_summary_df.head(),
    label="tab:summary table",
    caption="Classifier Statistics Summary"
)
Latex(final_summary_table)
```

Table 3: Classifier Statistics Summary

Model	TP	FP	TN	FN	accuracy	TPR	TNR
Linear Kernel SVM	33	3	32	2	0.93	0.94	0.91
Gaussian Kernel SVM	34	2	33	1	0.96	0.97	0.94
Polynomial Kernel SVM	32	2	33	3	0.93	0.91	0.94
k-Nearest Neighbor Classifier	34	2	33	1	0.96	0.97	0.94
K-Means Clustering Classifier	4	2	38	0	0.95	1.00	0.95

References

[1] How to Plot K-Means Clusters with Python? - AskPython — askpython.com. **https://www.askpython.com/python/examples/plot-k-means-clusters-python**. [Accessed 23-Apr-2023].

[2] José Fernando Costa. pandas: Find column with min/max value for each row in dataframe — medium.com. **https://medium.com/nerd-for-tech/pandas-find-column-with-min-max-value-for-each-row-in-dataframe-a2f2d2b2ea7a**. [Accessed 23-Apr-2023].

[3] Imad Dabbura. K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks — towardsdatascience.com. **https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a**. [Accessed 23-Apr-2023].

[4] Introduction to k-Means Clustering with scikit-learn in Python — datacamp.com. **https://www.datacamp.com/tutorial/k-means-clustering-python**. [Accessed 23-Apr-2023].

[5] Elbow Method for optimal value of k in KMeans - GeeksforGeeks — geeksforgeeks.org. **https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/**. [Accessed 21-Apr-2023].

[6] Grouping data points with k-means clustering. — jeremyjordan.me.

https://www.jeremyjordan.me/grouping-data-points-with-k-means-clustering/. [Accessed 24-Apr-2023].

[7] Wamika Jha. Implementation of Principal Component Analysis(PCA) in K Means Clustering — medium.com. **https://medium.com/analytics-vidhya/implementation-of-principal-component-analysis-pca-in-k-means-clustering-b4bc0aa79cb6**. [Accessed 23-Apr-2023].

[8] Knee/Elbow Point Detection — kaggle.com. **https://www.kaggle.com/code/kevinarvai/knee-elbow-point-detection**. [Accessed 21-Apr-2023].

[9] Mudassir Khan. KMeans Clustering for Classification — towardsdatascience.com. **https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a**. [Accessed 21-Apr-2023].

[10] Lorraine Li. K-Means Clustering with scikit-learn — towardsdatascience.com. **https://towardsdatascience.com/k-means-clustering-with-scikit-learn-6b47a369a83c**. [Accessed 24-Apr-2023].

[11] How to use knee point detection in k means clustering — practicaldatascience.co.uk. **https://practicaldatascience.co.uk/machine-learning/how-to-use-knee-point-detection-in-k-means-clustering**. [Accessed 21-Apr-2023].

[12] kmeans clustering centroid - Python — pythonprogramminglanguage.com. **https://pythonprogramminglanguage.com/kmeans-clustering-centroid/**. [Accessed 23-Apr-2023].

[13] Arif R. Step by Step to Understanding K-means Clustering and Implementation with sklearn — medium.com. **https://medium.com/data-folks-indonesia/step-by-step-to-understanding-k-means-clustering-and-implementation-with-sklearn-b55803f51** [Accessed 23-Apr-2023].

[14] Konstantin Rink. Four mistakes in Clustering you should avoid — towardsdatascience.com. **https://towardsdatascience.com/common-mistakes-in-cluster-analysis-and-how-to-avoid-them-eb960116d773**. [Accessed 23-Apr-2023].

[15] Lea Setruk. K-Means Algorithm (with example) — leasetruk.medium.com. **https://leasetruk.medium.com/k-means-algorithm-with-example-b8994b3c4ef3**. [Accessed 21-Apr-2023].

[16] Natasha Sharma. K-Means Clustering Explained — neptune.ai. **https://neptune.ai/blog/k-means-clustering**. [Accessed 23-Apr-2023].

[17] 2.3. Clustering — scikit-learn.org. **https://scikit-learn.org/stable/modules/clustering.html**. [Accessed 21-Apr-2023].

[18] sklearn.cluster.KMeans — scikit-learn.org. **https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html**. [Accessed 20-Apr-2023].

[19] 1.4. Support Vector Machines — scikit-learn.org. **https://scikit-learn.org/stable/modules/svm.html**. [Accessed 20-Apr-2023].

[20] sklearn.svm.SVC — scikit-learn.org. **https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html**. [Accessed 20-Apr-2023].