TEAM ID : H24 MEMBER : Yujin kim, ykim68

ALDA Fall 2025
HW2
Due: 09/17/2025

HW2 contains 3 questions. Please read and follow the instructions.

- **DUE DATE FOR SUBMISSION: 09/17/2025 11:45 PM**
- **TOTAL NUMBER OF POINTS: 100**
- **NO PARTIAL CREDIT** will be given so provide concise answers.
- **You MUST manually add ALL team members in the submission portal when you submit through Gradescope.** One submission per group. Team member(s) who are left out will lose 20 points if added after the deadline.
- Make sure you clearly list **your homework team ID, all team members' names and Unity IDs, for those who have contributed to the homework contribution** at the top of your submission.

https://github.com/ykim68ncstate/ncsu-engr-ALDA-Fall2025-H24

TEAM ID : H24 MEMBER : Yujin kim, ykim68

[GradeScope and GitHub Submission Instructions for Coding Questions]: For coding questions, you must submit your written solutions as a PDF on GradeScope and also provide your code in a private GitHub repository with instructor access.

1. **Create a private GitHub repository for your group.** Name your repository using the following format:

   `ncsu-engr-ALDA-Fall2025-HXX`

   Replace `XX` with your homework group number. **Example:** `ncsu-engr-ALDA-Fall2025-H2`. You will use one homework repository for the whole semester: do not create a new repository for each homework.

2. **Set up the repository:**
   - Log in to GitHub and click the green **"New"** button.
   - Use the required naming convention.
   - Set the repository to **private**.
   - Click **"Create repository"**.

3. **Add collaborators:**
   - Navigate to **Settings** > **Collaborators**.
   - Add all group members.
   - **Add the instructors and TAs.**

4. **Organize your code:**
   - Create separate folders for each of five assignments (e.g., `HW1`, `HW2`, etc.).
   - Place all relevant code in the correct folder.
   - Upload your complete `.py` files **before the submission deadline**. Late or missing code will not be graded.

5. **Include your GitHub repository link in the GradeScope PDF.**

6. **Clearly reference your code in the PDF:**
   - Indicate the relevant file for each question.
     Example: "The solution to Question 2 is in `matrix.py`."
   - Code must be executable; outputs must be generated by running the file.
   - **Do not hardcode results.** Submissions without proper computation will receive no credit.

7. **Code excerpts and outputs:** Your PDF must include output for each part and key code snippets (not full scripts) that demonstrate your implementation logic.

---

https://github.com/ykim68ncstate/ncsu-engr-ALDA-Fall2025-H24

1. (40 points) [**Coding - PCA**] [**Graded by Ian Holmes**] PCA is an unsupervised learning algorithm which uses an orthogonal transformation to convert data to new dimensions. In this problem, you will perform a PCA on the provided training dataset ("pca_train.csv") and the testing dataset ("pca_test.csv"). In both datasets, each row represents a data point or sample and the last column "Class" is a feature which indicate a class for each sample. The rest of the columns are input features.
Follow the instructions under "GradeScope and GitHub Submission Instructions for Coding Questions" (see page 2). Write code in Python to perform the following tasks.You may use the following libraries: NumPy, Pandas, Scikit-learn, and Matplotlib to solve this problem.
Note: You must include your output and the relevant key codes in the pdf submitted through Gradescope and must upload the original .py code on NCSU GitHub before the submission deadline.
   (a) (2 points) Load the data. Report the size of the training and testing sets. How many Class (1) and Class (0) samples are in the training set and the testing set, respectively? Test (46, 61) - Class 0 : 24, Class 1 : 22 / Train (146, 61) - Class 0 : 78, Class 1 : 68
   (b) (18 points) **Preprocessing Data-Normalization**: Please run normalization on all input features in both the training and testing datasets to obtain the *normalized* training and the *normalized* testing datasets. (**Hint:** you need to use the *min/max of each column* in the training dataset to normalize the testing dataset, and do NOT normalize the output "Class" of data.)

      Use the **NEW** normalized datasets for the following tasks :

      i. (2 points) Calculate the covariance matrix of the *NEW* training dataset. Please 1) specify the dimension of the resulted covariance matrix and 2) given the space limitation, please report the first $5 * 5$ of the covariance matrix, that is, only reporting the first five rows and the first five columns of the entire covariance matrix. Cov matrix shape : (60, 60) , first 5x5 → in Code①
      ii. (2 points) Calculate the eigenvalues and the eigenvectors based on the entire covariance matrix in (i) above. Report the size of the covariance matrix and the 5 largest eigenvalues. Cov matrix shape : (60×60), 5 eigenvalue : [0.3156 0.2629 0.1193 0.0977 0.095]
      iii. (1 point) Display the eigenvalues using a bar graph or a plot, and choose a reasonable number(s) of eigenvectors. Justify your answer.
      iv. (13 points) Next, you will combine PCA with a *K-nearest neighbor (KNN)* classifier. More specifically, PCA will be applied to reduce the dimensionality of data by transforming the original data into p ($p \leq 60$) principal components; and then KNN ($K = 5$, euclidean distance as distance metric) will be employed to the p principal components for classification (third-party packages are allowed to use for KNN).
         – (5 points) Report the accuracy of the *NEW* testing dataset when using PCA ($p = 4$) with 5NN. To show your work, please submit the corresponding csv file (including the name of csv file in your answer below).

iii) bar graph → in code ② (p=10)
The eigenvalues decrease sharply for the first few principal components and then gradually decline, forming an elbow pattern. Therefore, selecting 10 principal components, where the decrease starts to stabilize, seems reasonable. This choice avoid unnecessary dimensionality and improves computational efficiency

↳ file name : [b.iv_normalization_pca_knn_p4_results.csv]

Your csv file should have 6 columns: columns 1-4 are the 4 principal components, column 5 is the original ground truth output "Class", and the last column is the *predicted* output "Class". Test accuracy : 0.7391

- (6 points) Plot your results by varying p: 2, 4, 8, 10, 20, 40 and 60 respectively. In your plot, the x-axis represents the number of principal components and the y-axis refers to the accuracy of the *NEW* testing dataset using the corresponding number of principal components and 5NN. → in code ③
- (2 point) Based upon the (PCA + 5NN)'s results above, what is the **most "reasonable" number** of principal components among all the choices? Justify your answer.

(c) (18 points) **Preprocess Data-Standardization**: Similarly, please run standardization on all input features to obtain the *standardized* training and the *standardized* testing datasets. Then repeat the four steps i-iv in (b) above on the two **NEW** *standardized* datasets.

(d) (2 points) Comparing the results from (b) and (c), which of the two data-processing procedures, normalization or standardization, would you prefer for the given datasets? And why? (Answer without any justification will get zero point.)

(b)-iv

: Accuracy increase steadily and sharply between p=2 and p=20, reaching its highest value at p=20. After p=20, the accuracy slightly decreases and shows only minor fluctuations. Therefore, p appears to be the most reasonable number of principal components, as it achieves the highest accuracy while avoiding unnecessary dimensionality increase and reducing computational cost.

(c)-i Cov matrix shape : (60, 60) , first 5x5 → in code ④

(c)-ii Cov matrix shape : (60x60), 5 eigenvalue : [10.1145 9.7552 3.8864 3.2785 2.9077]

(c)-iii bar graph → in code ⑤ (p=10)

: The eigenvalue plot for the standardized dataset shows a steep drop in the first few principal components, followed by an elbow pattern. A reasonable choice of principal components is around 10, as selecting more than that would lead to unnecessary dimensionality increase and higher computational cost without substantial performance gain.

(c)-iv Testing accuracy : 0.7609

Filename : [c.iv_standardization_pca_knn_p4_results.csv]

plot → in code ⑥

: The graph shows a sharp increase in accuracy from 2 to 6 principal components, followed by a gradual but consistent improvement up to 40 components, the accuracy remains nearly unchanged. Therefore, using 40 principal components appears to be the most reasonable choice, as it achieves high accuracy without unnecessary dimensionality increase.

(d) Based on the normalization and standardization procedures applied to the current dataset, normalization is preferred. This is because normalization achieves an accuracy of 0.8696 at p=20, which is higher than the 0.8043 accuracy at p=40 obtained with standardization, while requiring fewer principal components. Therefore, normalization can improve computational efficiency for this dataset.

(a) Load the data.

Report the size of the training and testing sets. How many Class (1) and Class (0) samples are in the training set and the testing set, respectively?

```
import pandas as pd

PCA_test = pd.read_csv("HW2_data/pca_test.csv")
PCA_train = pd.read_csv("HW2_data/pca_train.csv")

print(PCA_test.shape)
print(PCA_train.shape)

print('Test:')
print(PCA_test['Class'].value_counts())
print('Train:')
print(PCA_train['Class'].value_counts())

print(PCA_test.describe())
print(PCA_train.head())
```

(b) Preprocessing Data-Normalization:

Please run normalization on all input features in both the training and testing datasets to obtain the normalized training and the normalized testing datasets. (Hint: you need to use the min/max of each column in the training dataset to normalize the testing dataset, and do NOT normalize the output "Class" of data.)

```
features = PCA_train.drop(columns=['Class']).columns

train_min = PCA_train[features].min()
train_max = PCA_train[features].max()

train_normalization = (PCA_train[features] - train_min) / (train_max - train_min)
test_normalization = (PCA_test[features] - train_min) / (train_max - train_min)

train_normalization['Class'] = PCA_train['Class']
test_normalization['Class'] = PCA_test['Class']

print("NaN: ",train_normalization.isna().any().any() or test_normalization.isna().any().any())
print("Inf:", train_normalization.isin([float('inf'), float('-inf')]).any().any()) or (test_normalization.isin([float('inf'), float('-inf')]).any().any())

print(train_normalization.describe())
```

i. Calculate the covariance matrix of the NEW training dataset.

Please 1) specify the dimension of the resulted covariance matrix and 2) given the space limitation, please report the first 5 * 5 of the five rows and the first five columns of the entire covariance matrix.

```
import numpy as np

train_features_only = train_normalization.drop(columns=['Class'])
cov_matrix = np.cov(train_features_only, rowvar=False)

print("Covariance matrix shape:", cov_matrix.shape)
print("First 5 * 5:")
print(cov_matrix[:5,:5])
```

✓ 0.0s

```
Covariance matrix shape: (60, 60)
First 5 * 5:
[[0.03667374 0.01993076 0.01566491 0.00845833 0.00884005]
 [0.01993076 0.02420038 0.01708202 0.01082221 0.00960302]
 [0.01566491 0.01708202 0.01970539 0.01241038 0.01275864]
 [0.00845833 0.01082221 0.01241038 0.01581789 0.01367937]
 [0.00884005 0.00960302 0.01275864 0.01367937 0.02206249]]
```

} code ①

ii. Calculate the eigenvalues and the eigenvectors based on the entire covariance matrix in (i) above.

Report the size of the covariance matrix and the 5 largest eigenvalues.

```
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
#https://numpy.org/doc/2.1/reference/generated/numpy.linalg.eigh.html#

idx = np.argsort(eigenvalues)[::-1]
eigenvalues_sorted = eigenvalues[idx]
eigenvectors_sorted = eigenvectors[:, idx]

print("Covariance matrix shape:", cov_matrix.shape)
print("Eigenvalues count:", eigenvalues_sorted.shape[0])
print("5 largest eigenvalues:", eigenvalues_sorted[:5])
```

✓ 0.0s

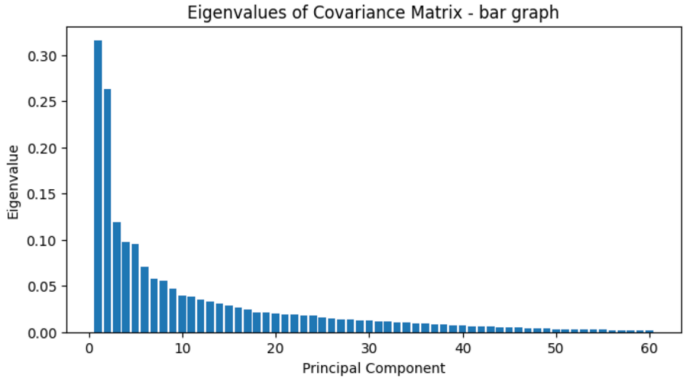
iii. Display the eigenvalues using a bar graph or a plot, and choose a reasonable number(s) of eigenvectors.

Justify your answer.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.bar(range(1, len(eigenvalues_sorted) + 1), eigenvalues_sorted)
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.title('Eigenvalues of Covariance Matrix - bar graph')
plt.show()
```

✓ 0.2s



← code ②

iv. Next, you will combine PCA with a K-nearest neighbor (KNN) classifier.

More specifically, PCA will be applied to reduce the dimensionality of data by transforming the original data into p (p ≤ 60) principal components; and then KN components for classification.

Report the accuracy of the NEW testing dataset when using PCA (p = 4) with 5NN. To show your work, please submit the corresponding csv file (including the are the 4 principal components, column 5 is the original ground truth output "Class", and the last column is the predicted output "Class".

```
from sklearn.neighbors import KNeighborsClassifier

Xtrain = train_normalization.drop(columns=['Class']).values
Ytrain = train_normalization['Class'].values
Xtest = test_normalization.drop(columns=['Class']).values
Ytest = test_normalization['Class'].values

def project_with_p(X, W, p):
    W_p = W[:, :p]
    return X @ W_p

def knn_acc_for_p(p):
    Xtrain_p = project_with_p(Xtrain, eigenvectors_sorted, p)
    Xtest_p = project_with_p(Xtest, eigenvectors_sorted, p)
    knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
    knn.fit(Xtrain_p, Ytrain)
    prediction = knn.predict(Xtest_p)
    acc = (prediction == Ytest).mean()
    return acc, prediction, Xtest_p


p = 4
acc_p4, prediction_p4, Xtest_p4 = knn_acc_for_p(p)

p4_df = pd.DataFrame(
    np.column_stack([Xtest_p4, Ytest, prediction_p4]),
    columns=[f"PC{i+1}" for i in range(p)] + ["True_Class", "Predicted_Class"]
)
csv_filename = "b.iv_normalization_pca_knn_p4_results.csv"
p4_df.to_csv(csv_filename, index=False)

print(f"[p=4] Test Accuracy: {acc_p4:.4f}")
print(f"Saved CSV: {csv_filename}")
```

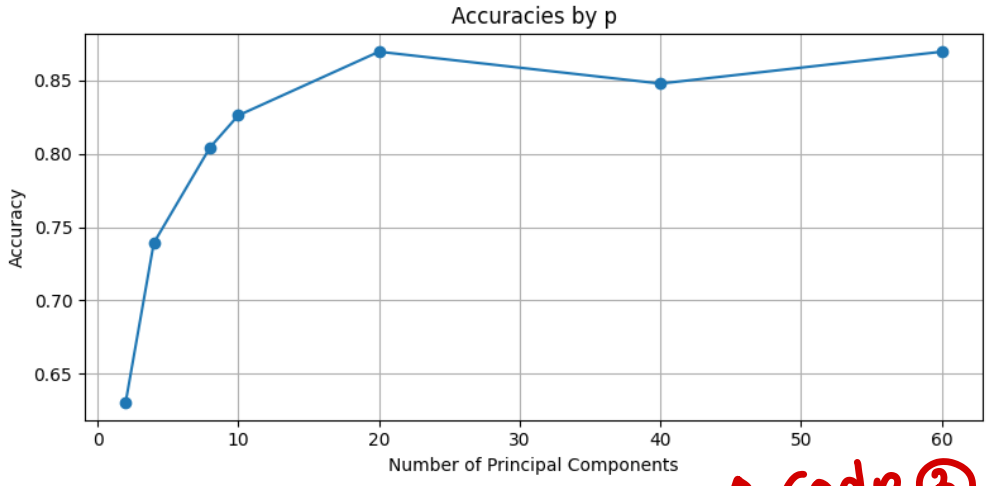
```python
p_values = [2, 4, 8, 10, 20, 40, 60]
accuracies = []
for pv in p_values:
    acc, _, _ = knn_acc_for_p(pv)
    accuracies.append(acc)

print("Accuracies by p:", list(zip(p_values, accuracies)))

plt.figure(figsize = (8, 4))
plt.plot(p_values, accuracies, marker='o')
plt.xlabel('Number of Principal Components')
plt.ylabel('Accuracy')
plt.title('Accuracies by p')
plt.grid(True)
plt.tight_layout()
plt.show()
```

✓ 0.0s

```
Accuracies by p: [(2, np.float64(0.6304347826086957)), (4, np.float64(0.7391304347826086)), (8, np.float64(0.804347
```



▲ Code ③

(c) Preprocess Data-Standardization:

Similarly, please run standardization on all input features to obtain the standardized training and the standardized testing datasets. Then repeat the four steps i-iv in (b) above on the two NE

```python
features = PCA_train.drop(columns=['Class']).columns
train_mean = PCA_train[features].mean()
train_standardization = PCA_train[features].std(ddof=0)

train_standardization_df = (PCA_train[features] - train_mean) / train_standardization
test_standardization_df = (PCA_test[features] - train_mean) / train_standardization

train_standardization_df['Class'] = PCA_train['Class'].values
test_standardization_df['Class'] = PCA_test['Class'].values

print("NaN: ",train_standardization_df.isna().any().any() or test_standardization_df.isna().any().any())
print("Inf:", train_standardization_df.isin([float('inf'), float('-inf')]).any().any()) or (test_standardization_df.isin([float('inf'), float('-inf')]).any().any())

print(train_standardization_df.describe())
```

(c) i. Calculate the covariance matrix of the NEW training dataset.

Please 1) specify the dimension of the resulted covariance matrix and 2) given the space limitation, please report the first 5 * 5 of the covarian matrix.

```
train_features_only_standardization = train_standardization_df.drop(columns=['Class'])
cov_matrix_standardization = np.cov(train_features_only_standardization, rowvar=False)

print("Covariance matrix shape of Standardization:", cov_matrix_standardization.shape)
print("First 5 * 5:")
print(cov_matrix_standardization[:5,:5])
```

✓ 0.0s

```
Covariance matrix shape of Standardization: (60, 60)
First 5 * 5:
[[1.00689655 0.67362775 0.58673595 0.35360459 0.31292083]
 [0.67362775 1.00689655 0.78762676 0.5569491  0.41846036]
 [0.58673595 0.78762676 1.00689655 0.70778774 0.61612573]
 [0.35360459 0.5569491  0.70778774 1.00689655 0.73730849]
 [0.31292083 0.41846036 0.61612573 0.73730849 1.00689655]]
```

code ④

(c) ii. Calculate the eigenvalues and the eigenvectors based on the

Report the size of the covariance matrix and the 5 largest eigenvalues.

```
eigenvalues_standardization, eigenvectors_standardization = np.linalg.eigh(cov_matrix_standardization)

idx = np.argsort(eigenvalues_standardization)[::-1]
eigenvalues_standardization_sorted = eigenvalues_standardization[idx]
eigenvectors_standardization_sorted = eigenvectors_standardization[:, idx]

print("Covariance matrix shape of standardization:", cov_matrix_standardization.shape)
print("Eigenvalues count:", eigenvalues_standardization_sorted.shape[0])
print("5 largest eigenvalues:", eigenvalues_standardization_sorted[:5])
```

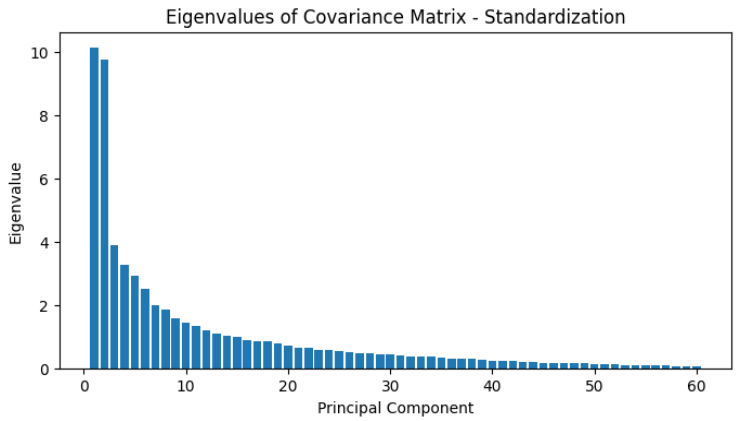
(c) iii. Display the eigenvalues using a bar graph or a plot, and choose a

Justify your answer.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.bar(range(1, len(eigenvalues_standardization_sorted) + 1), eigenvalues_standardization_sorted)
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.title('Eigenvalues of Covariance Matrix - Standardization')
plt.show()
```

✓ 0.0s



← code ⑤

(c) iv. Next, you will combine PCA with a K-nearest neighbor (KNN) classifier.

More specifically, PCA will be applied to reduce the dimensionality of data by transforming the original data into p (p ≤ 60) principal components; and then KNN (K = 5 components for classification.

Report the accuracy of the NEW testing dataset when using PCA (p = 4) with 5NN. To show your work, please submit the corresponding csv file (including the name c are the 4 principal components, column 5 is the original ground truth output "Class", and the last column is the predicted output "Class".

```
Xtrain_standardization = train_standardization_df.drop(columns=['Class']).values
Ytrain_standardization = train_standardization_df['Class'].values
Xtest_standardization = test_standardization_df.drop(columns=['Class']).values
Ytest_standardization = test_standardization_df['Class'].values

def project_with_p_standardization(X, W, p):
    W_p = W[:, :p]
    return X @ W_p

def knn_acc_for_p_standardization(p):
    Xtrain_p_standardization = project_with_p_standardization(Xtrain_standardization, eigenvectors_standardization_sorted, p)
    Xtest_p_standardization = project_with_p_standardization(Xtest_standardization, eigenvectors_standardization_sorted, p)
    knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
    knn.fit(Xtrain_p_standardization, Ytrain_standardization)
    prediction = knn.predict(Xtest_p_standardization)
    acc = (prediction == Ytest_standardization).mean()
    return acc, prediction, Xtest_p_standardization


p = 4
acc_p4_standardization, prediction_p4_standardization, Xtest_p4_standardization = knn_acc_for_p_standardization(p)

p4_standardization_df = pd.DataFrame(
    np.column_stack([Xtest_p4_standardization, Ytest_standardization, prediction_p4_standardization]),
    columns=[f"PC{i+1}" for i in range(p)] + ["True_Class", "Predicted_Class"]
)
csv_filename_standardization = "c.iv_standardization_pca_knn_p4_results.csv"
p4_standardization_df.to_csv(csv_filename_standardization, index=False)

print(f"[standardization,p=4] Test Accuracy: {acc_p4_standardization:.4f}")
print(f"Saved CSV: {csv_filename_standardization}")
```

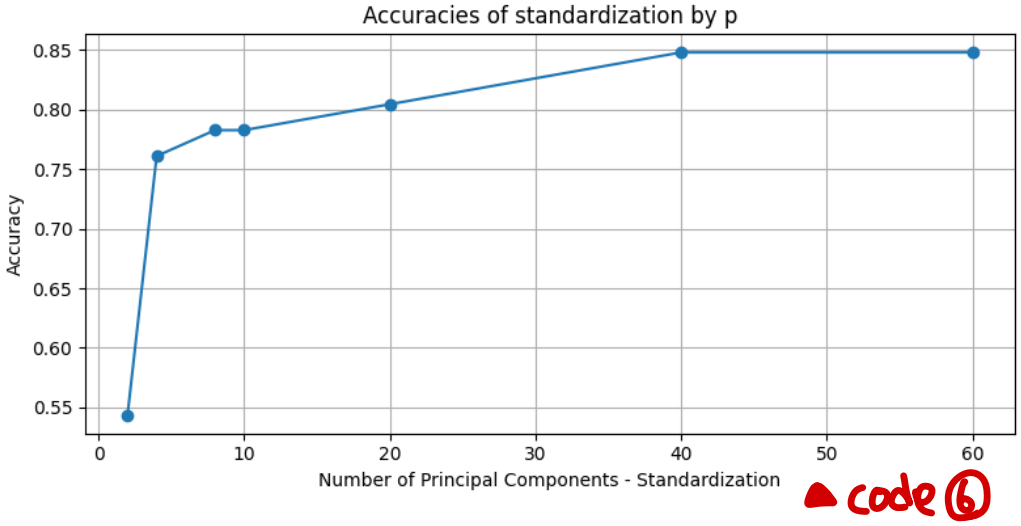
```python
p_values = [2, 4, 8, 10, 20, 40, 60]
accuracies_standardization = []
for pv_standardization in p_values:
    acc_standardization, _, _ = knn_acc_for_p_standardization(pv_standardization)
    accuracies_standardization.append(acc_standardization)

print("Accuracies of standardization by p:", list(zip(p_values, accuracies_standardization)))

plt.figure(figsize = (8, 4))
plt.plot(p_values, accuracies_standardization, marker='o')
plt.xlabel('Number of Principal Components - Standardization')
plt.ylabel('Accuracy')
plt.title('Accuracies of standardization by p')
plt.grid(True)
plt.tight_layout()
plt.show()
```

✓ 0.0s

```
Accuracies of standardization by p: [(2, np.float64(0.5434782608695652)), (4, np.float64(0.7608695652173914)), (8, n
```



code 6

2. (30 points) [**Decision Tree (Very Time-consuming)**] [**Graded by Tural Mehtiyev**] For this exercise, you will use `weather_forecast_data.csv`, which contains 12 weather forecasts from various atmospheric conditions.

   The output label *forecast* has the following values:

   1. **Sunny**: The forecast was sunny.
   2. **Rainy**: The forecast was rainy.

   The attributes can be described as:

   1. Sky Condition: clear, cloudy
   2. Temperature Band: cold, mild, hot, extreme
   3. Humidity: high, low
   4. Wind Direction: N, S, E, W

   Complete the following tasks using the decision tree algorithm discussed in the lecture. Note that multiple-way splitting is allowed. *In the case of ties, break ties in favor of the leftmost attribute.* If you find that splitting a node cannot reduce its entropy, then you can leave it unsplit. (You can hand-draw all of your trees on paper and scan your results into the final PDF.)

   (a) (15 points) Construct the tree *manually* using ID3/entropy computations. Write down the computation process by showing the number of cases in each class for each node before splitting and show your tree step by step. (No partial credit)

   (b) (15 points) Construct the tree *manually* using the GINI index, take the attribute *Sky Condition* as the root (then generate the next two levels of the remaining tree by choosing the best split), write down the computation process by showing the number of cases in each class for each node before splitting and show your tree step by step. (No partial credit)

103 / Entropy

1. Caculate Entropy

1) Class Ratio

Sun = 7 → $P_S = 7/12 = 0.5833$

Rain = 5 → $P_R = 5/12 = 0.4167$

$$H(Fo) = \sum_{i=1}^{k} -P_i \log_2 P_i$$

k = 2 (Sun / Rain)

$$= -(0.5833 \log_2 0.5833 + 0.4167 \log_2 0.4167)$$

$$= -(0.5833 \times (-0.78) + 0.4167 \times (-1.27))$$

$$= 0.455 + 0.529 = 0.984$$

$H(Fo = Sun) = 7/12$

$H(Fo) = 0.98$

Label

sky_condition	temperature_band	humidity	wind_direction	forecast
Cloudy	mild	low	W	rain
clear	hot	low	W	sun
clear	extreme	high	E	sun
cloudy	mild	low	S	sun
clear	hot	low	W	rain
cloudy	mild	low	E	sun
clear	cold	low	W	rain
clear	cold	high	N	rain
Cloudy	mild	high	S	sun
cloudy	hot	high	E	sun
cloudy	hot	low	E	sun
clear	hot	high	E	rain

① Sky_condition

Clear : 6 (Sun = 2, Rain = 4)

$H = -\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6}\right) = 0.9183$

Cloudy : 6 (Sun = 5, Rain = 1)

$H = -\left(\frac{5}{6}\log_2\frac{5}{6} + \frac{1}{6}\log_2\frac{1}{6}\right) = 0.65$

$H(Fo|S) = \frac{6}{12} \times 0.9183 + \frac{6}{12} \times 0.65 = 0.78$

② Temperature_band

mild : 4 (Sun = 3, Rain = 1)

$H = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.81$

hot : 5 (Sun = 3 , Rain = 2)

$H = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) = 0.97$

cold : 2 (Sun = 0, Rain = 2)

$H = 0$

extreme : 1 (Sun = 1, Rain = 0)

$H = 0$

$H(Fo|T) = \frac{4}{12} \times 0.81 + \frac{5}{12} \times 0.97 = 0.67$

③ humidity

low : 7 (Sun = 4 , Rain = 3)

$H = -\left(\frac{4}{7}\log_2\frac{4}{7} + \frac{3}{7}\log_2\frac{3}{7}\right) = 0.985$

high : 5 (Sun = 4, Rain = 1)

$H = -\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 0.97$

$H(Fo|H) = \frac{7}{12} \times 0.985 + \frac{5}{12} \times 0.97 = 0.98$

④ Wind_direction

W : 4 (Sun = 1 , Rain = 3)

$H = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) = 0.81$

S : 2 (Sun = 2 , Rain = 0)

$H = 0$

E : 5 (Sun = 4 , Rain = 1)

$H = -\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 0.72$

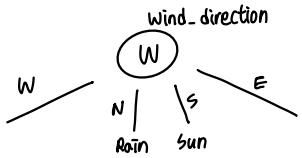
N : 1 (Sun = 0 , Rain = 1)

$H = 0$

$H(Fo|W) = \frac{4}{12} \times 0.81 + \frac{5}{12} \times 0.72 = 0.57$

* $H(Fo) = 0.98$

$H(Fo\|S) = 0.78$	$IG(Fo\|S) = 0.98 - 0.78 = 0.2$
$H(Fo\|T) = 0.67$	$IG(Fo\|T) = 0.98 - 0.67 = 0.31$
$H(Fo\|H) = 0.97875$	$IG(Fo\|H) = 0.98 - 0.97875 = 0.00125$
$H(Fo\|W) = 0.57$	$IG(Fo\|W) = 0.98 - 0.57 = 0.41$



ID3 / Entropy

2) Sub branch : Wind_direction [W]

$P(Fo = Sun) = 1/4$ $H(Fo) = 0.81$

① Sky_condition

clear : 3 (Sun = 1, Rain = 2)

$H = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = 0.918$

Cloudy : 1 (Sun=0, Rain=1)

$H = 0$

$\frac{3}{4} \times 0.918 = 0.69$

② Temperature_band

mild : 1 (Sun = 0, Rain = 1) → H=0

cold : 1 (Sun=0, Rain=1) → H=0

hot : 2 (Sun=1, Rain=1)

$H = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$

$\frac{2}{4} \times 1 = 0.5$

③ humidity

low : 4 (Sun = 1, Rain = 3)

$H = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) = 0.81$

$H(Fo) = 0.81$

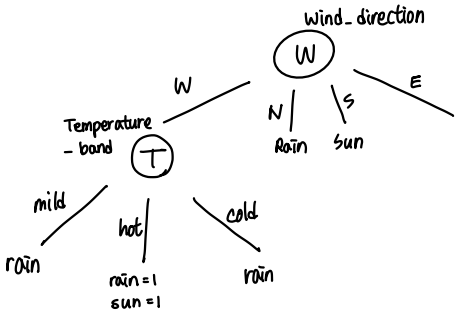
$H(Fo|S) = 0.69$ $IG(Fo|S) = 0.81 - 0.69 = 0.12$

$H(Fo|T) = 0.5$ $IG(Fo|T) = 0.81 - 0.5 = 0.31$

$H(Fo|H) = 0.81$ $IG(Fo|H) = 0.81 - 0.81 = 0$

Label

sky_condition	temperature_band	humidity	wind_direction	forecast
Cloudy	mild	low	W	rain
clear	hot	low	W	sun
clear	hot	low	W	rain
clear	cold	low	W	rain



sky_condition	temperature_band	humidity	wind_direction	forecast
clear	hot	low	W	sun
clear	hot	low	W	rain

$P(Fo|Sun) = 1/2$ $H(Fo) = 1$

① Sky_condition

clear : 2 (Sun=1, rain=1)

$H = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$

② Humidity

low : 2 (Sun=1, rain=1)

$H = 1$

$H(Fo|S) = 1$ $IG(Fo|S) = 0$

$H(Fo|H) = 1$ $IG(Fo|H) = 0$

103/Entropy

3.Sub branch : Wind_direction [E]

$P(Fo = Sun) = 4/5$ $H(Fo) = 0.72$

① Sky_condition

clear: 2 (Sun = 1, Rain = 1)

$H = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1$

Cloudy : 3 (Sun = 3, Rain = 0)

$H = 0$

$H(Fo|S) = \frac{2}{5} \times 1 = 0.4$

② Temperature_band

extreme : 1 (Sun = 1, Rain = 0)

$H = 0$

mild : 1 (Sun = 1, Rain = 0)

$H = 0$

hot : 3 (Sun = 2, Rain = 1)

$H = -(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}) = 0.918$

$H(Fo|T) = \frac{3}{5} \times 0.918 = 0.55$

③ humidity

high : 3 (Sun = 2, Rain = 1)

$H = -(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}) = 0.918$

low : 2 (Sun = 2, Rain = 0)

$H = 0$

$H(Fo|H) = \frac{3}{5} \times 0.918 = 0.55$

$H(Fo) = 0.72$

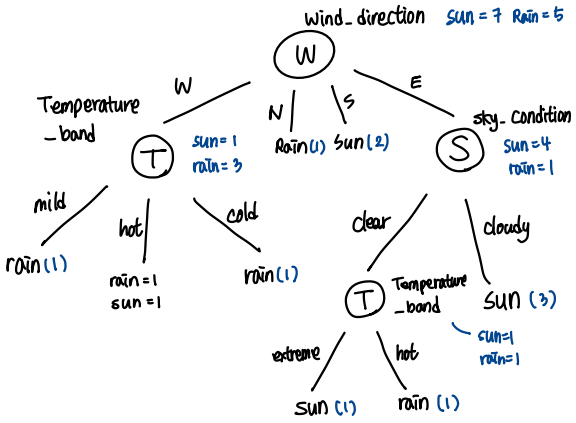
$H(Fo|S) = 0.4$ $IG(Fo|S) = 0.72 - 0.4 = 0.32$

$H(Fo|T) = 0.55$ $IG(Fo|T) = 0.72 - 0.55 = 0.17$

$H(Fo|H) = 0.55$ $IG(Fo|H) = 0.72 - 0.55 = 0.17$

Label

sky_condition	temperature_band	humidity	wind_direction	forecast
clear	extreme	high	E	sun
cloudy	mild	low	E	sun
cloudy	hot	high	E	sun
cloudy	hot	low	E	sun
clear	hot	high	E	rain



Output

sky_condition	temperature_band	humidity	wind_direction	forecast
clear	extreme	high	E	sun
clear	hot	high	E	rain

$P(Fo = Sun) = 1/2$ $H(Fo) = 1$

① temperature_band

extreme : 1 (Sun = 1, Rain = 0) → H = 0

hot : 1 (Sun = 0, Rain = 1) → H = 0

$H(Fo|T) = 0$

② humidity

high : 2 (Sun = 1, Rain = 1)

$H = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1$

$H(Fo|H) = 1$

$IG(Fo|T) = 1$, $IG(Fo|H) = 0$

GINI

	Parent
Sun	7
Rain	5

$GINI(t) = 1 - \sum [p(j|t)]^2$

$= 1 - \left(\frac{7}{12}\right)^2 - \left(\frac{5}{12}\right)^2$

$= 1 - 0.34 - 0.17 = 0.49$

(fix) Sky_condition (clear : 6, Cloudy : 6)

	clear
sun	2
Rain	4

$P(sun) = \frac{2}{6}$ $P(Rain) = \frac{4}{6}$

$Gini = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2$

$= 1 - 0.11 - 0.44 = 0.45$

	cloudy
sun	5
Rain	1

$P(sun) = \frac{5}{6}$ $P(Rain) = \frac{1}{6}$

$Gini = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2$

$= 1 - 0.694 - 0.028 = 0.278$

$G = \frac{6}{12} \times 0.45 + \frac{6}{12} \times 0.278 = 0.36$

Label

sky_condition	temperature_band	humidity	wind_direction	forecast
Cloudy	mild	low	W	rain
clear	hot	low	W	sun
clear	extreme	high	E	sun
cloudy	mild	low	S	sun
clear	hot	low	W	rain
cloudy	mild	low	E	sun
clear	cold	low	W	rain
clear	cold	high	N	rain
Cloudy	mild	high	S	sun
cloudy	hot	high	E	sun
cloudy	hot	low	E	sun
clear	hot	high	E	rain

② Second-level : clear

sky_condition	temperature_band	humidity	wind_direction	forecast
clear	hot	low	W	sun
clear	extreme	high	E	sun
clear	hot	low	W	rain
clear	cold	low	W	rain
clear	cold	high	N	rain
clear	hot	high	E	rain

* temperature_band

	hot
sun	1
Rain	2

$P(sun) = \frac{1}{3}$ $P(Rain) = \frac{2}{3}$

$Gini = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2$

$= 1 - 0.11 - 0.44 = 0.45$

	cold
sun	0
Rain	2

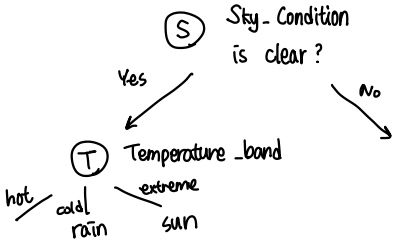
$P(sun) = 0$ $P(Rain) = 1$

$Gini = 1 - 0^2 - 1^2 = 0$

	extreme
sun	1
Rain	0

$Gini = 0$

$G = \frac{3}{6} \times 0.45 = 0.225$



* humidity

	low
sun	1
Rain	2

$P(sun) = \frac{1}{3}$ $P(Rain) = \frac{2}{3}$

$Gini = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.45$

	high
sun	1
Rain	2

$P(sun) = \frac{1}{3}$ $P(Rain) = \frac{2}{3}$

$Gini = 0.45$

$G = \frac{3}{6} \times 0.45 + \frac{3}{6} \times 0.45 = 0.45$ (x split)

* wind_direction

	W
S	1
R	2

$P(S) = \frac{1}{3}$ $P(R) = \frac{2}{3}$

$Gini = 0.45$

	E
S	1
R	1

$P(S) = \frac{1}{2}$ $P(R) = \frac{1}{2}$

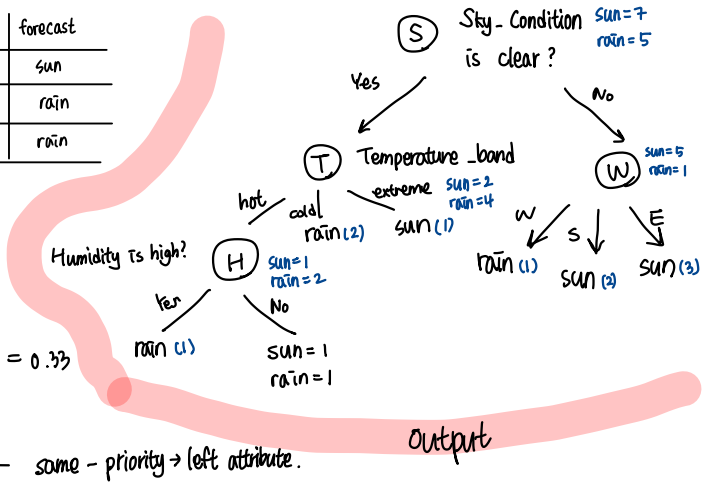
$Gini = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$

	N
S	0
R	1

$Gini = 0$

$G = \frac{3}{6} \times 0.45 + \frac{2}{6} \times 0.5 = 0.39$

sky_condition	temperature_band	humidity	wind_direction	forecast
clear	hot	low	W	sun
clear	hot	low	W	rain
clear	hot	high	E	rain



* humidity

	low
S	1
R	1

$P(S) = 1/2 \quad P(R) = 1/2$

$Gini = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

	high
S	0
R	1

$Gini = 0$

$G = \frac{2}{3} \times 0.5 = 0.33$

same - priority → left attribute.
⇒ humidity

* wind_direction

	W
S	1
R	1

$Gini = 0.5$

$G = \frac{2}{3} \times 0.5 = 0.33$

	E
S	0
R	1

$Gini = 0$

③ second-level : cloudy

Label

sky_condition	temperature_band	humidity	wind_direction	forecast
cloudy	mild	low	W	rain
cloudy	mild	low	S	sun
cloudy	mild	low	E	sun
cloudy	mild	high	S	sun
cloudy	hot	high	E	sun
cloudy	hot	low	E	sun

* temperature_band

	mild
S	3
R	1

$P(S) = \frac{3}{4} \quad P(R) = \frac{1}{4}$

$Gini = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2$

$= 1 - 0.56 - 0.06 = 0.38$

	hot
S	2
R	0

$Gini = 0$

$G = \frac{4}{6} \times 0.38 = 0.253$

* humidity

	low
S	3
R	1

$Gini = 0.38$

	high
S	2
R	0

$Gini = 0$

$G = 0.253$

* wind_direction

	W
S	0
R	1

$Gini = 0$

	S
S	2
R	0

$Gini = 0$

$G = 0$

	E
S	2
R	0

$Gini = 0$

3. (30 points) [**Evaluate Classifier**] [**Graded by Rajesh Debnath**] Bob is very superstitious, and wants to predict how his daily choices control whether or not he has a lucky day. To improve his predictions, Bob decides to create his own decision tree to help him determine whether his day will be lucky (LUCKY if it's a lucky day, NORMAL if it's a normal day). The decision tree Bob created is shown in the Figure below.

   Your task is to determine whether to prune the given decision tree. Additionally, you will use the provided test dataset in "luckyday.csv" to assess the effectiveness of the resulting decision trees.

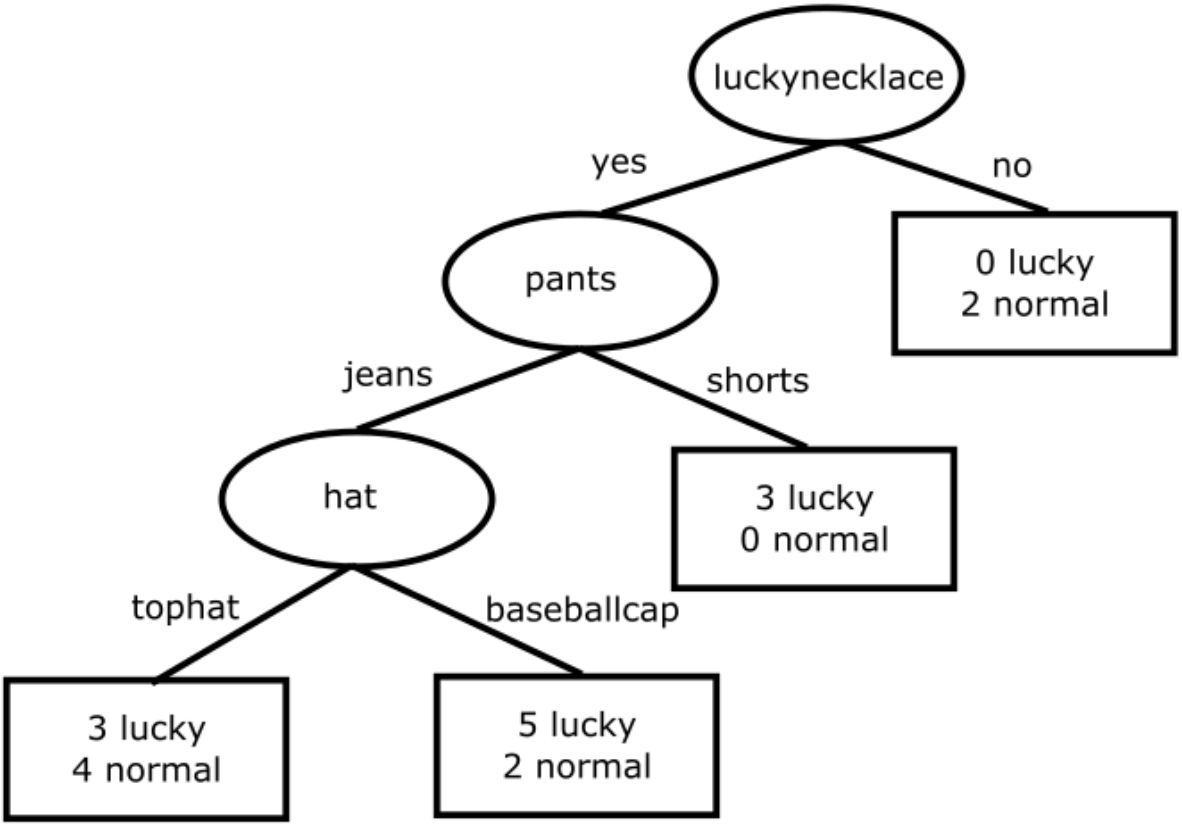


Figure 1: Lucky day decision tree.

   (a) (10 points) Post-pruning based on **optimistic errors**.
      i. (3 points) Calculate the optimistic errors before splitting and after splitting using the attribute *hat* respectively.
      ii. (2 points) Based upon the optimistic errors, would the subtree be pruned or retained? If it is pruned, draw the resulting decision tree and use it for the next question; otherwise, use the original decision tree shown in Figure 1 for the next question.
      iii. (5 points) Use the decision tree from (a).ii above to classify the test dataset (hw2_q3.csv). Report its performance on the following five eval-

uation metrics: Accuracy, Recall (Sensitivity), Precision, Specificity, and F1 Measure.

(b) (10 points) Post-pruning based on **pessimistic errors**. When calculating pessimistic errors, each leaf node will add a factor of **2** to the error.

i. (3 points) Calculate the pessimistic errors before splitting and after splitting using the attribute *hat* respectively.

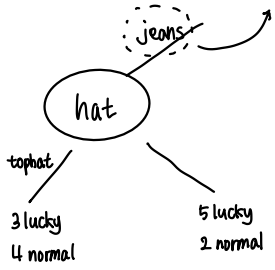
ii. (2 points) Based on the pessimistic errors, would the subtree be pruned or retained? If it is pruned, draw the resulting decision tree and use it for the next question; otherwise, use the original decision tree shown in Figure 13 for the next question.

iii. (5 points) Use the decision tree from (b).ii above to classify the test dataset (hw2_q3.csv). Report its corresponding five evaluation metrics: Accuracy, Recall(Sensitivity), Precision, Specificity, and F1 Measure.

(c) (10 points) We will compare the performance of the decision trees from (a).ii and from (b).ii using the test dataset (hw2_q3.csv). Bob says that the most important thing to him is to never incorrectly predict a lucky day: it's better to have a model that accidentally predicts a normal day when it's really a lucky day, than to have a model that accidentally predicts a lucky day when it's really a normal day. For the task of predicting whether Bob will have a lucky day or not, which of the five evaluation metrics: Accuracy, Recall(Sensitivity), Precision, Specificity, and F1 Measure, are the most important? Based on your selected evaluation metrics, which decision tree, (a).ii or (b).ii, is better for this task? Justify your answers.

(a) - i)

jeans { luckyday = 8, normal = 6



Before split - Optimistic Error

$E_{before} = \frac{6}{14} = 0.4286$

After split - Optimistic Error

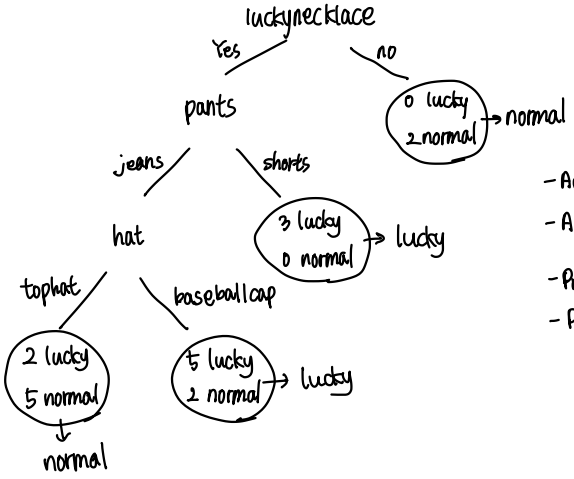
$E_{after} = \frac{3+2}{14} = \frac{5}{14} = 0.3571$

(a) - ii)

$E_{after}(0.36) < E_{before}(0.43)$ (x prune)

→ keep split (original decision tree)

(a) - iii)



- Actual LUC = 10
- Actual NOM = 9
- Predict LUC = 10
- Predict NOM = 9

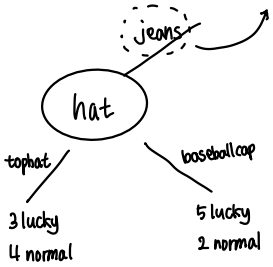
Act	Pre	
N	N	TN
N	N	TN
L	L	TP
L	L	TP
L	L	TP
L	N	FN
L	N	FN
N	N	TN
N	N	TN
N	N	TN
N	N	TN
N	N	TN
L	L	TP
L	L	TP
L	L	TP
L	L	TP
L	L	TP
N	L	FP
N	L	FP

confusion matrix : TP / FN / FP / TN

	Predict LUC	Predict NOM
Actual LUC	TP 8	FN 2
Actual NOM	FP 2	TN 7

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{8+7}{19} = \frac{15}{19} = 0.789$
- Recall = $\frac{TP}{TP+FN} = \frac{8}{10} = 0.8$
- Precision = $\frac{TP}{TP+FP} = \frac{8}{10} = 0.8$
- Specify = $\frac{TN}{TN+FP} = \frac{7}{9} = 0.78$
- F1-Score = $2x \frac{Precision \times Recall}{Precision + Recall} = 2x \frac{0.8 \times 0.8}{0.8+0.8} = \frac{1.28}{1.6} = 0.8$

(b) - i)



jeans { luckyday = 8, normal = 6

Before Split - Pessimistic Error (panelty = 2)

$E_{before} = \frac{6+2}{14} = 0.57$

After Split - Pessimistic Error (panelty = 2)

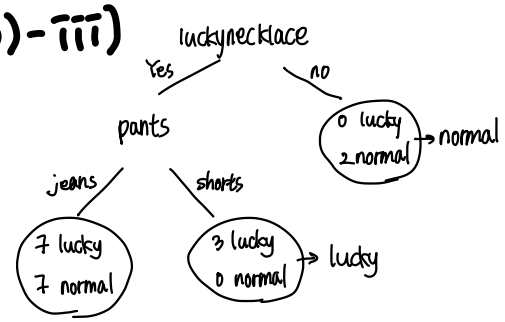
$E_{after} = \frac{5+2\times2}{14} = 0.64$

(b) - ii)

$E_{after}(0.64) > E_{before}(0.57)$

→ prune

(b) - iii)



↓ ← follow parent leaf

lucky

- Actual LUC = 10
- Actual NOM = 9
- Predict LUC = 17
- Predict NOM = 2

Act	Pre	
N	N	TN
N	N	TN
L	L	TP
L	L	TP
L	L	TP
L	L	TP
L	L	TP
N	L	FP
N	L	FP
N	L	FP
N	L	FP
N	L	FP
L	L	TP
L	L	TP
L	L	TP
L	L	TP
L	L	TP
N	L	FP
N	L	FP

- Confusion matrix

	Predict LUC	Predict NOM
Actual LUC	TP 10	FN 0
Actual NOM	FP 7	TN 2

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{10+2}{19} = \frac{12}{19} = 0.63$
- Recall = $\frac{TP}{TP+FN} = \frac{10}{10+0} = 1$
- Precision = $\frac{TP}{TP+FP} = \frac{10}{10+7} = \frac{10}{17} = 0.588$
- Specify = $\frac{TN}{TN+FP} = \frac{2}{2+7} = \frac{2}{9} = 0.22$
- F1-Score = $2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2\times0.588\times1}{0.588+1} = 0.74$

(c)

	(a)		(b)
Accuracy	0.789	>	0.63
Recall	0.8	<	1
Precision	0.8	>	0.588
specify	0.78	>	0.22
F1-Score	0.8	>	0.74

For this task, [Precision] and [Specificity] are most important metric because Bob wants to avoid falsely predicting a lucky day (FP). [Precision] directly measures how many of the predicted lucky days are actually lucky, and [Specificity] measures how well the model correctly identifies normal days.

Comparing the two decision trees, the tree from (a) has significantly higher [Precision] and [Specificity] than the tree (b), as it produces 5 fewer FP out of only 19 samples. Therefore, the decision tree from (a) is better for this task, since it minimize FP predictions and aligns with Bob's priority.