تمرین چهارم درس داده کاوی **محمدباربد امیرمزلقانی – ۸۱۰۱۰۲۳۴۸**

بخش تشریحی

سوال اول)

ابتدا فرمول آنتروپی را مینویسیم که به صورت مقابل است :

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

که n تعداد کلاس هاست و p(i) احتمال هر کدام.

حال در کل ۹ تا بله داریم و ۵ تا خیر که p آن ها به صورت $\frac{9}{14}$ و $\frac{5}{14}$ میشود حال برای محاسبه آنتروپی طبق فرمول عمل میکنیم و حاصل این عبارت را بدست میاوریم :

$$\mathrm{H}(S) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right)$$

در نهایت با انجام محاسبات این مقدار ۰.۹۴۴۲ بدست میاید.

سوال دوم)

این مقدار از تفاوت آنتروپی اصلی و جمع وزن دار آنتروپی به ازای یک فیچر (مثل دما) بدست میاید.

از فرمول مانند قبل استفاده میکنیم برای دما سه حالت گرم، معتدل و خنک داریم حال برای هر یک از سه حالت آنتروپی را حساب میکنیم مثلا برای گرم دو تا خیر و یک بله داریم برای معتدل چهار بله و دو خیر داریم و برای خنک چهار بله و یک خیر داریم.

و فرمول محاسبه هر کدام را در ادامه میبینیم :

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

$$H(Mild) = -\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) \approx 0.918$$

$$H(Cool) = -\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) \approx 0.722$$

حال اختلاف میانگین وزن دار این سه مقدار را محاسبه میکنیم:

$$H_{\text{subsets}} = \frac{3}{14} \cdot 0.918 + \frac{6}{14} \cdot 0.918 + \frac{5}{14} \cdot 0.722 \approx 0.656$$

که در نهایت اختلاف آن را حساب میکنیم :

$$IG(S,\text{Temperature}) = H(S) - H_{\text{subsets}} = 0.944 - 0.656 \approx 0.288$$

سوال سوم)

این الگوریتم ویژگی با بالاترین information gain را به عنوان روت درخت قرار میدهد پس لازم است IG را به ازای تک تک آن ها حساب کنیم دیگر فرمول به ازای تک تک حالات را نمینویسیم(در بالا هست) و مقدار آنها را قرار میدهیم که به صورت زیر است :

باد :

ضعیف : ۰.۸۱۱ ، قوی : ۰.۷۲۲

که میانگین وزن دار آن برابر ۰.۷۸۸ شد و IG آن برابر ۰.۱۵۶.

رطوبت :

زیاد : ۰.۷۲۲، نرمال : ۰.۸۱۱

که میانگین وزن دار آن برابر ۰.۷۵۹ شد و IG برابر ۰.۱۸۵.

وضعیت :

آفتابی : ۰.۹۱۸، ابری : ۰، بارانی : ۰.۷۲۲

میانگین وزن دار آنها برابر ۰.۶۹۴ شد و IG آن برابر ۰.۲۵

در نهایت بزرگترین IG مربوط به دما بود با مقدار ۰.۲۸۸ که به عنوان ریشه درخت قرار میگیرد.

سوال چهارم)

با توجه به مرحله قبل ریشه ما دما شد حال دیتاست را بر مبنای آن تقسیم میکنیم و سپس بهترین را دوباره بر مبنای IG برای هر قسمت پیدا میکنیم.

که سه دسته ما به صورت زیر هستند

گرم با سه عضو(۲ خیر و ۱ بله)، معندل با شش عضو(۴ بله و ۲ خیر) و خنک با پنج عضو (۴ بله و ۱ خیر)

آنتروپی هیچکدام هم صفر نیست پس لازم است ادامه دهیم .

برای هر کدام دوباره طبق الگوریتم که بالاتر توضیح دادیم پیش میرویم و بدین صورت میشود برای گرم بالاترین IG را باد پیدا میکند که به ازای باد ضعیف خیر و باد قوی خیر هستیم.

برای معتدل رطوبت بهترین است که رطوبت بالا خیر و رطوبت پایین بله است و در نهایت برای خنک چشم انداز بهترین است که به ازای آفتابی بارانی و یا ابری جواب بله است.

سوال پنجم)

حال در این قسمت از درختی که در بخش قبل ساختیم و توضیح دادیم استفاده میکنیم :

برای اولی ابتدا دما را چک میکنیم که گرم است پس در مرحله بعد باید دما چک شود که ضعیف است پس جواب پیش بینی ما خیر است.

برای حالت دوم هم که ابتدا هوا خنک است پس وارد شاخه سوم میشویم و باید چشم انداز را چک کنیم، چشم انداز آفتابی است پس جواب ما بله است.

سوال ششم)

با توجه به اینکه یکی را درست تشخیص داد و یک را غلط، اگر معیار Accuracy باشد که به صورت تعداد صحیح بر کل تعدادهاست : برابر ۰.۵ یا ۵۰٪ میشود.

سوال هفتم)

تعداد هفت قانون به صورت زیر میتوان بدست آورد :

اگر هوا گرم باشد و باد ضعیف آنگاه برگزار نمیشود.

اگر هوا گرم باشد و باد قوی آنگاه برگزار نمیشود.

اگر هوا معتدل باشد و رطوبت زیاد باشد آنگاه برگزار نمیشود.

اگر هوا معتدل باشد و رطوبت عادی باشد آنگاه برگزار میشود.

اگر هوا خنک باشد و چشم انداز ابری باشد آنگاه برگزار میشود.

اگر هوا خنک باشد و چشم انداز آفتابی باشد انگاه برگزار میشود.

اگر هوا خنک باشد و چشم انداز بارانی باشد آنگاه برگزار میشود.

بخش عملی)

بخش اول)

ابتدا به بخش پیش پردازش میپردازیم در همین بخش باید چندین گام را اجرا کنیم که در ادامه آن را میبینیم:

الف)چک کردن مقادیر و از دست رفته ها :

```
import pandas as pd

file_path = 'Question_1.csv'
data = pd.read_csv(file_path)

unique_values = data.nunique()
missing_values = data.isnull().sum()

unique_values, missing_values
```

```
(Unnamed: 0                61069
 cap-diameter               2571
 cap-shape                     7
 cap-surface                  11
 cap-color                    12
 does-bruise-or-bleed          2
 gill-attachment               7
 gill-spacing                  3
 gill-color                   12
 stem-height                2226
 stem-width                 4630
 stem-root                     5
 stem-surface                  8
 stem-color                   13
 veil-type                     1
 veil-color                    6
 has-ring                      2
 ring-type                     8
 spore-print-color             7
 habitat                       8
 season                        4
 class                         2
 dtype: int64,
 Unnamed: 0                    0
 cap-diameter                  0
 cap-shape                     0
```

برای این بخش از فانکشن های bulit-in مربوط به pandas استفاده کردیم.

ب) در ادامه این استراتژی را انتخاب میکنیم که ابتدا اگر بیش از ۵۰ درصد گم شده بودند آن ستون حذف شود، سپس داده های عددی و غیرعددی را جدا میکنیم برای داده های عددی از میانگین و برای غیرعددی ها مد را جایگزین میکنیم که کد آن به صورت زیر میشود:

```python
from sklearn.impute import SimpleImputer

threshold = len(data) * 0.5
data_cleaned = data.dropna(thresh=threshold, axis=1)

numerical_cols = data_cleaned.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = data_cleaned.select_dtypes(include=['object']).columns

num_imputer = SimpleImputer(strategy='mean')
data_cleaned[numerical_cols] = num_imputer.fit_transform(data_cleaned[numerical_cols])

cat_imputer = SimpleImputer(strategy='most_frequent')
data_cleaned[categorical_cols] = cat_imputer.fit_transform(data_cleaned[categorical_cols])
```

ج)تبدیل ویژگی های غیر عددی به عددی خواسته بعدی سوال برای اینکار از Label Encoding استفاده میکنیم که خودش یک مدل فیت میکند و مقادیر عددی میشوند:

```python
from sklearn.preprocessing import LabelEncoder

label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    data_cleaned[col] = le.fit_transform(data_cleaned[col])
    label_encoders[col] = le
```

د) در این قسمت نرمالایزیشن را داریم که انتخاب روش به عهده خودمان بود و من از StandardScaler استفاده کردم که به نحو زیر است و head آن را هم جهت راحت شدن فهم نشان دادم:

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data_cleaned[numerical_cols] = scaler.fit_transform(data_cleaned[numerical_cols])

data_cleaned.head()
```

```
<ipython-input-6-3f4e6ed4f262>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  data_cleaned[numerical_cols] = scaler.fit_transform(data_cleaned[numerical_cols])
```

	Unnamed: 0	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	
0	-1.732022	1.619462	6	2	6	0	2	0	10	3.
1	-1.731966	1.873982	6	2	6	0	2	0	10	3
2	-1.731909	1.393432	6	2	6	0	2	0	10	3.
3	-1.731852	1.412426	2	3	1	0	2	0	10	2.
4	-1.731796	1.501699	6	3	6	0	2	0	10	2.

ه) در این قسمت یک کار مهم باید انجام دهیم که چک کردن توازن مجموع دادست، در همین راستا ابتدا چک میکنیم در فیچر مقصد چه تعداد از هر کلاس داریم که با توجه به نتایج بدست آمده ناتوازنی به حدی نیست که لازم باشد نمونه گیری انجام دهیم :

```
class_counts = data_cleaned['class'].value_counts()
class_counts
```

```
class
1    33888
0    27181
Name: count, dtype: int64
```

حال به سراغ آموزش مدل ها طبق خواسته سوال میرویم که آن هم مراحل زیر را دارد

الف) در مرحله اول طبق خواسته سوال داده را به نسبت ۰.۸ ۰.۲ تقسیم میکنیم :

```
from sklearn.model_selection import train_test_split

X = data_cleaned.drop('class', axis=1)
y = data_cleaned['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

ب-ج-د-ه-و-ز) در این قسمت لازم است تا یک درخت تصمیم به کمک SK آموزش دهیم و دقت آن را روی تست حساب کنیم همچنین لازم است کارایی و زمان اجرا را هم برای ترین هم برای تست حساب کنیم :

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import time

start_train = time.time()
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
end_train = time.time()

start_predict = time.time()
y_pred = clf.predict(X_test)
end_predict = time.time()

train_time = end_train - start_train
predict_time = end_predict - start_predict

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)

accuracy, precision, recall, f1, train_time, predict_time, conf_matrix
```

```
(0.9970525626330441,
 0.9970526968140749,
 0.9970525626330441,
 0.9970523855642777,
 0.8507506847381592,
 0.011942863464355469,
 array([[5353,   21],
        [  15, 6825]]))
```

در بخش بعد همین مراحل را داریم ولی به جای درخت تصمیم از KNN استفاده میکنیم که کد و نتایج آن به صورت زیر میشود :

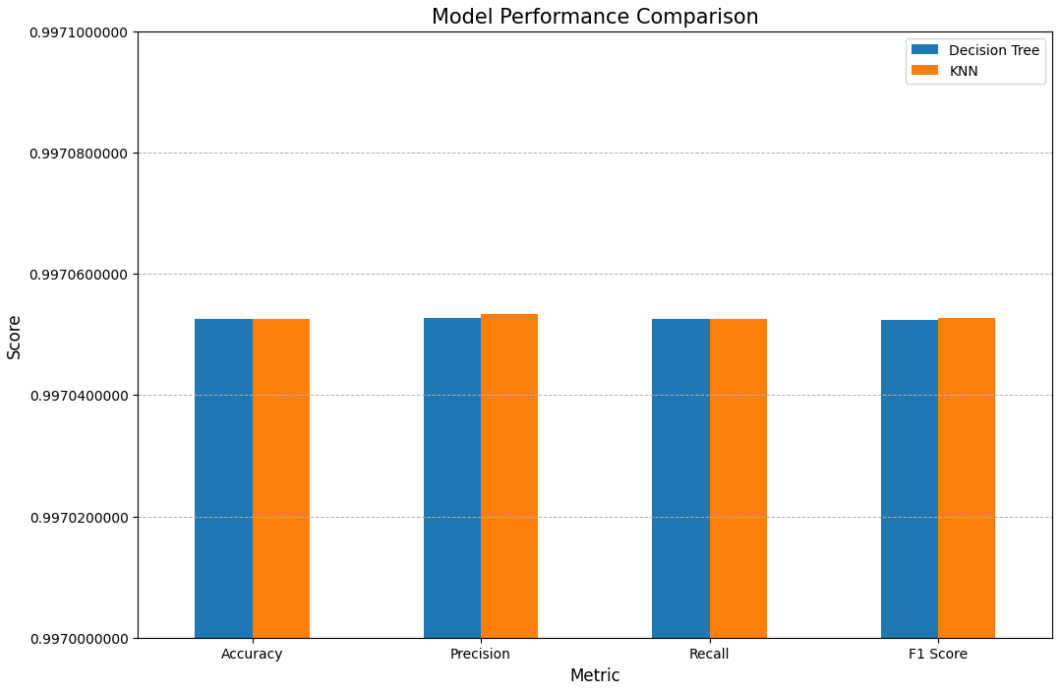
```python
from sklearn.neighbors import KNeighborsClassifier

start_train_knn = time.time()
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train)
end_train_knn = time.time()

start_predict_knn = time.time()
y_pred_knn = knn.predict(X_test)
end_predict_knn = time.time()

train_time_knn = end_train_knn - start_train_knn
predict_time_knn = end_predict_knn - start_predict_knn

accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn, average='weighted')
recall_knn = recall_score(y_test, y_pred_knn, average='weighted')
f1_knn = f1_score(y_test, y_pred_knn, average='weighted')
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)

accuracy_knn, precision_knn, recall_knn, f1_knn, train_time_knn, predict_time_knn, conf_matrix_knn
```

```
(0.9970525626330441,
 0.9970534018853338,
 0.9970525626330441,
 0.9970527382162941,
 0.02963995933532715,
 6.727258682250977,
 array([[5359,   15],
        [  21, 6819]]))
```

در ادامه کد مربوط به نمودار میله ای را داریم که خروجی آن هم مشخص شده است :

```
[12] import pandas as pd

     metrics_df = pd.DataFrame({
         'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score'],
         'Decision Tree': [accuracy, precision, recall, f1],
         'KNN': [accuracy_knn, precision_knn, recall_knn, f1_knn]
     })
```

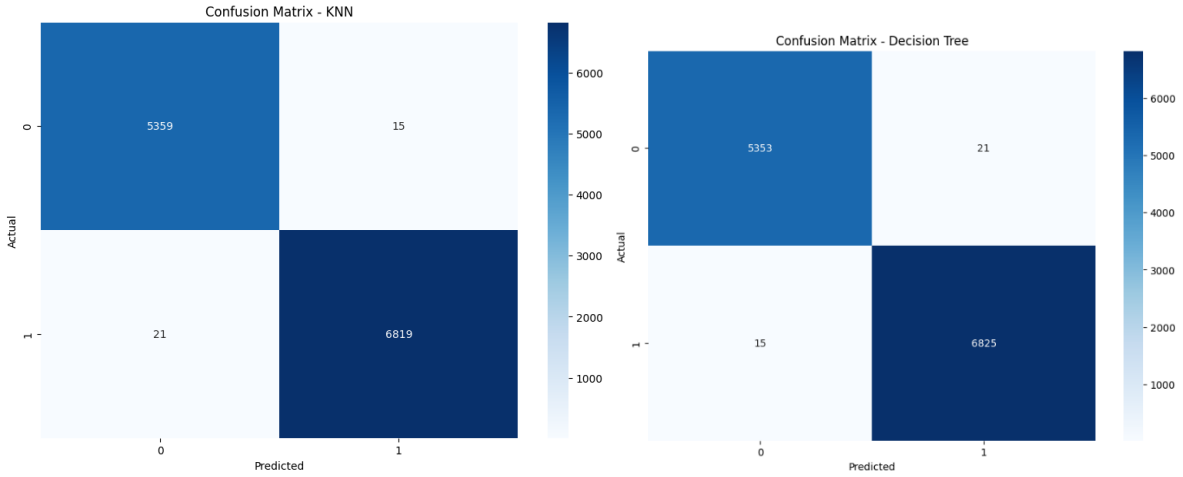
```
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots(figsize=(12, 8))

metrics_df.plot(x='Metric', kind='bar', ax=ax)

ax.yaxis.set_major_formatter(plt.FuncFormatter(lambda y, _: '{:.10f}'.format(y)))
ax.set_ylim(0.997, 0.9971)

plt.grid(axis='y', linestyle='--', linewidth=0.7)

plt.title('Model Performance Comparison', fontsize=15)
plt.ylabel('Score', fontsize=12)
plt.xlabel('Metric', fontsize=12)

plt.xticks(rotation=0)

plt.legend(loc='upper right')

plt.show()
```



که حدودا برابر هستند اما KNN اندکی بهتر عمل میکند.

برای ماتریس آشفتگی هم کد آن مانند حالت قبل است و چیزی که پرینت کرده بودیم را فقط میکشیم که به صورت زیر میشود :

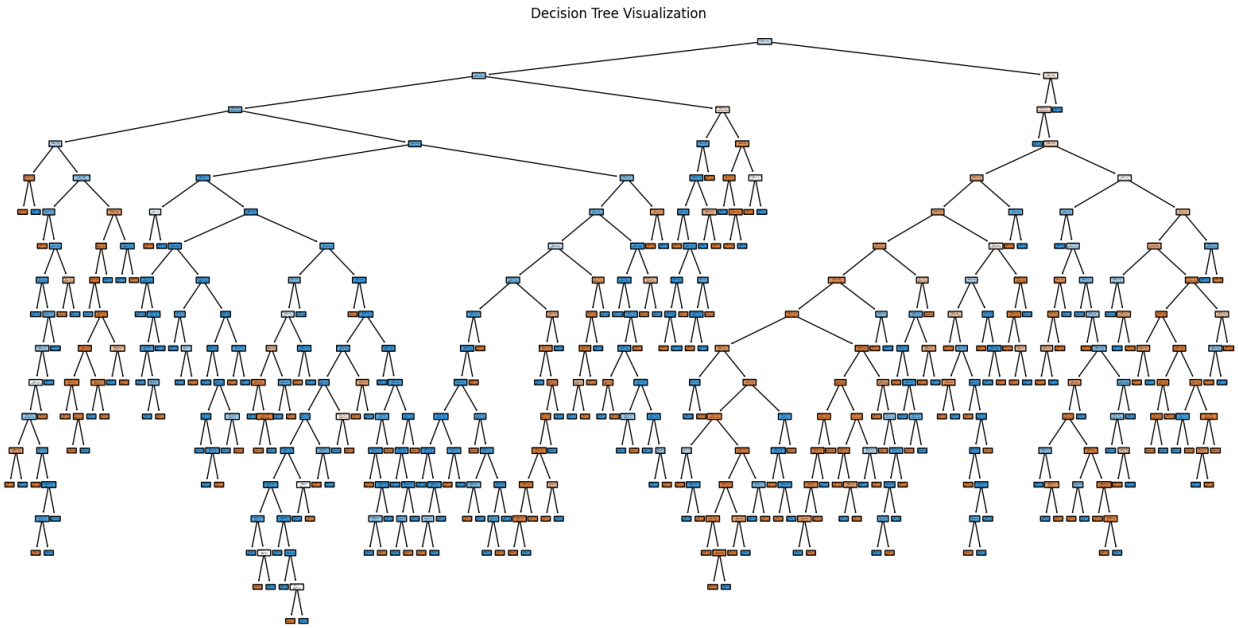
```
import seaborn as sns

plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Decision Tree')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix_knn, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - KNN')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



در تفسیر هم میتوان گفت true negative مشابه دارند با تفاوت جزئی در KNN که ۶ تا بهتر عمل کرده در true positive برعکس است یعنی درخت تصمیم ۶ تا بهتر عمل کرده . حال بستگی به هدف دارد که مثلا در بحث تشخیص کوچک کردن FN ها اولویت ماست پس درخت تصمیم مفیدتر است ولی مثلا در سیستمی مثل spam detectionکه هدف مینیمم کردن FP هاست اولویت ما KNN خواهد بود.

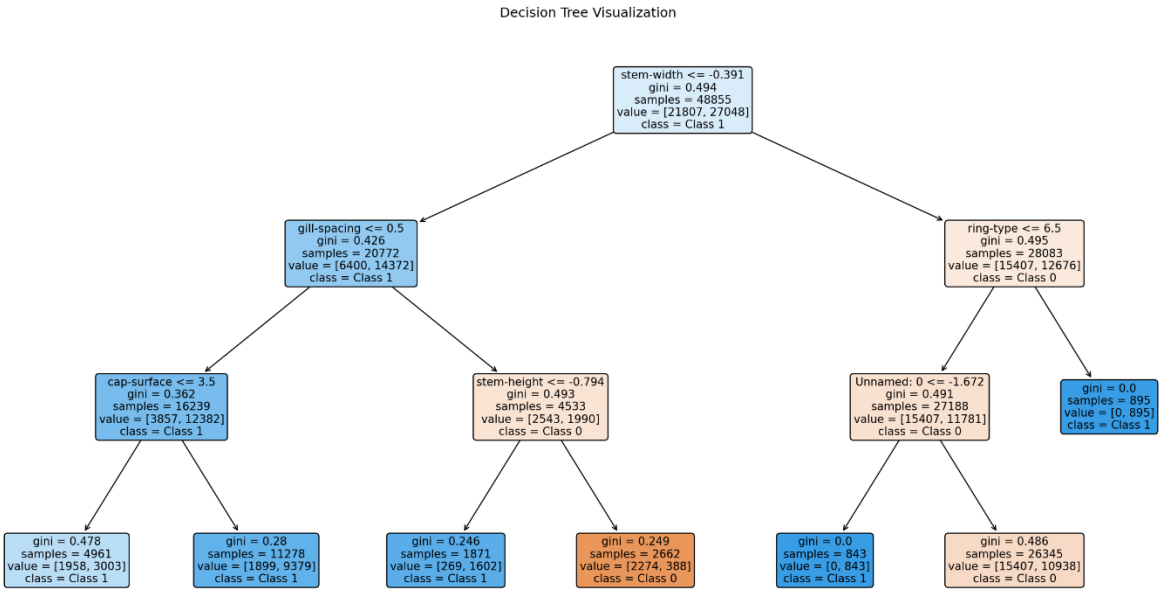
در نهایت هم نمایش درخت تصمیم را داریم که به کمک plot_tree آن را انجام میدهیم :

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

plt.figure(figsize=(20, 10))
plot_tree(clf, filled=True, feature_names=data_cleaned.drop('class', axis=1).column
plt.title('Decision Tree Visualization')
plt.show()
```



به ازای محدود کردن عمق میتوان درخت خواناتری بدست آورد که به صورت زیر میشود(همچنین میتوان فونت و DPI) را تغییر داد :

```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

clf = DecisionTreeClassifier(max_depth=3)
clf.fit(X_train, y_train)

plt.figure(figsize=(20, 10), dpi=150)
plot_tree(clf, filled=True, feature_names=data_cleaned.drop('class', axis=1).column
plt.title('Decision Tree Visualization')
plt.show()
```



بخش دوم)

الف) در این قسمت لازم است تا دنباله های دیتا را بسازیم با توجه به سایز ۱۰ که خواسته سوال است، ابتدا ۱۰ تا ۱۰ تا پیش میرویم و دیتا را به دو بخش sequence و labels تقسیم میکنیم :

```
[16] import numpy as np

     def create_sequences(data, window_size):
         sequences = []
         labels = []
         for i in range(len(data) - window_size):
             sequences.append(data[i:i + window_size])
             labels.append(data[i + window_size])
         return np.array(sequences), np.array(labels)

     window_size = 10
     sequences, labels = create_sequences(data['Temp'].values, window_size)

     sequences_df = pd.DataFrame(sequences, columns=[f'Day_{i+1}' for i in range(window_size)])
     labels_df = pd.DataFrame(labels, columns=['Day_11'])
```

ب) در این قسمت تنها کاری که لازم است انجام دهیم، تقسیم با نسبت ۰.۲ است که با کد زیر به راحتی انجام میگیرد :

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(sequences, labels, test_size=0.2, random_state=42)
```

ج)در این قسمت لازم است تا یک مدل رگرسیون بر دیتایی که داریم fit کنیم که به راحتی به کمک SK میتوان آن را انجام داد :

```
[19] from sklearn.linear_model import LinearRegression

     linear_model = LinearRegression()
     linear_model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

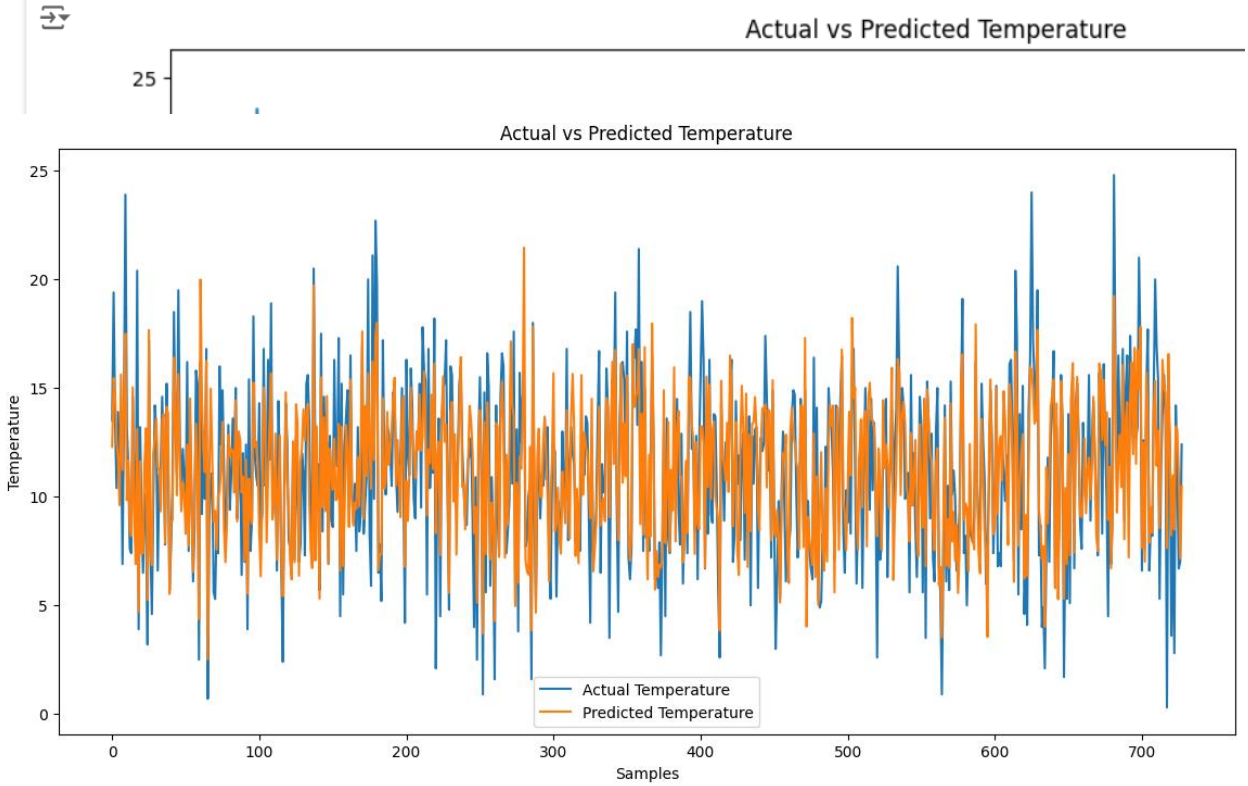
د) حال لازم است تا عملکرد مدل را بر روی دیتای تست بررسی کنیم و MAE و RMSE را را محاسبه کنیم که برای هر دو، دو فانکشن bulit-in داریم :

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

y_pred = linear_model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f'MAE: {mae}')
print(f'RMSE: {rmse}')
```

```
MAE: 1.9076905202696846
RMSE: 2.4113535227841543
```

ه) حال لازم است نمودار را برای داده تست، بین مقادیر واقعی و پیش بینی رسم کنیم که به کمک plt آن را رسم میکنیم :

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))
plt.plot(y_test, label='Actual Temperature')
plt.plot(y_pred, label='Predicted Temperature')
plt.xlabel('Samples')
plt.ylabel('Temperature')
plt.legend()
plt.title('Actual vs Predicted Temperature')
plt.show()
```



نمودار نشان می‌دهد که مدل رگرسیون خطی به طور کلی روند کلی تغییرات دما را دنبال می‌کند، اما در برخی از نقاط تفاوت قابل توجهی بین مقادیر واقعی و پیش‌بینی شده وجود دارد. به نظر می‌رسد که مدل در پیش‌بینی مقادیر دما در برخی از روزها موفق نبوده است و اختلاف‌های قابل ملاحظه‌ای بین مقادیر واقعی و پیش‌بینی شده دیده می‌شود.

نوسانات زیادی در داده‌ها وجود دارد و مدل رگرسیون خطی ممکن است نتواند به خوبی تمامی این نوسانات را پیش‌بینی کند.این مسئله نشان می‌دهد که مدل ممکن است در پیش‌بینی مقادیر با نوسانات بالا دچار مشکل شود.

با این حال، مدل توانسته است روند کلی تغییرات دما را تا حدی دنبال کند. در بسیاری از نقاط، منحنی پیش‌بینی شده نزدیک به منحنی واقعی است.این نشان می‌دهد که مدل می‌تواند برای پیش‌بینی روند کلی تغییرات دما مفید باشد، اما برای پیش‌بینی دقیق مقادیر نیاز به مدل‌های پیچیده‌تر و دقیق‌تری است.

استفاده از مدل‌های پیچیده‌تر مانند شبکه‌های عصبی بازگشتی (RNN) یا مدل‌های مبتنی بر تقویت‌سازی (Boosting)ممکن است دقت پیش‌بینی را افزایش دهد،استفاده از ویژگی‌های اضافی مانند اطلاعات بیشتر در مورد شرایط آب و هوایی می‌تواند به بهبود دقت مدل کمک کند، بهینه‌سازی هایپرپارامترهای مدل‌های فعلی نیز ممکن است عملکرد را بهبود بخشد.

و) تغییرات خواسته شده را اعمال میکنیم که لیبل تنها نشان دهنده کاهش یا افزایش باشد، در این راستا از خود label هم میتوانیم استفاده کنیم(چون دیتاها را شامل میشود) بدین صورت که اگر شرط برقرار بود ۱ قرار دهد در غیر این صورت برچسب ما صفر شود:

```
[23] new_labels = np.where(labels[1:] > labels[:-1], 1, 0)
     new_sequences = sequences[:-1]

     X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(new_sequences, new_labels, test_size=0.2, random_state=42)
```

ز)

حال یک مدل SVM بر روی داده های جدید فیت میکنیم که به صورت زیر است:

```
[24] from sklearn.svm import SVC

     svm_model = SVC(kernel='linear')
     svm_model.fit(X_train_new, y_train_new)
```

```
SVC
SVC(kernel='linear')
```

ح)در این قسمت تمامی معیارهای خواسته شده را برای داده تست بدست میاوریم :

```
[25] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

     y_pred_svm = svm_model.predict(X_test_new)

     accuracy = accuracy_score(y_test_new, y_pred_svm)
     precision = precision_score(y_test_new, y_pred_svm)
     recall = recall_score(y_test_new, y_pred_svm)
     f1 = f1_score(y_test_new, y_pred_svm)

     print(f'Accuracy: {accuracy}')
     print(f'Precision: {precision}')
     print(f'Recall: {recall}')
     print(f'F1 Score: {f1}')
```

```
Accuracy: 0.5851648351648352
Precision: 0.5995085995085995
Recall: 0.6370757180156658
F1 Score: 0.6177215189873417
```

ط) در این قسمت هدفمان پیدا کردن بهترین حدآستانه است به نحوی که توافق بین دو مدل به حداکثر برسد، در گام اول یک تابع برای محاسبه نرخ توافق بین دو مدل مینویسیم که عملکرد آن هم بدین صورت است که تک به تک چک میکند و در نهایت یک میانگین میگیرد.

در گام بعد هم سایز بودن آنها چک میشود و سپس یک رنجی از تمام حدآستانه ها چک میشود و بهترین آنها را به عنوان خروجی میدهیم:

```
import numpy as np

def calculate_agreement_rate(y_pred_linear_classified, y_pred_svm):
    return np.mean(y_pred_linear_classified == y_pred_svm)

if len(y_pred) > len(y_pred_svm):
    y_pred = y_pred[:len(y_pred_svm)]
elif len(y_pred_svm) > len(y_pred):
    y_pred_svm = y_pred_svm[:len(y_pred)]

best_threshold = 0
best_agreement_rate = 0

for threshold in np.arange(min(y_pred), max(y_pred), 0.01):
    y_pred_linear_classified = (y_pred > threshold).astype(int)
    agreement_rate = calculate_agreement_rate(y_pred_linear_classified, y_pred_svm)
    if agreement_rate > best_agreement_rate:
        best_agreement_rate = agreement_rate
        best_threshold = threshold

print(f'Best Threshold: {best_threshold}')
print(f'Best Agreement Rate: {best_agreement_rate}')
```

```
Best Threshold: 2.5114351435768425
Best Agreement Rate: 0.5604395604395604
```