دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق

مینی پروژه ۲ یادگیری ماشین

نگارش

رضا لاری

مهدی جوکار

استاد درس

دکتر علیاری

اردیبهشت ۱۴۰۴

لینک گوگل کولب:

سوال ۳:

https://colab.research.google.com/drive/1J58LWDCUgTzSAi9skhoh7ytfKCz9Rthg?authuser=3#scrollTo=MyigZlkWULsU

۳.پرسش سه- درخت تصمیم

۱.۳)

```
[48] #import numpy , matplotlib and seaborn

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

1.3- Read Data

```
[49] !gdown 1LARn0Boq62ijLSCKYnogBeCtLe2KSxZL
     company_data = pd.read_csv('Company_Data.csv')
     company_data.head(10)
```

```
Downloading...
From: https://drive.google.com/uc?id=1LARn0Boq62ijLSCKYnogBeCtLe2KSxZL
To: /content/Company_Data.csv
100% 16.6k/16.6k [00:00<00:00, 42.3MB/s]
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
5	10.81	124	113	13	501	72	Bad	78	16	No	Yes
6	6.63	115	105	0	45	108	Medium	71	15	Yes	No
7	11.85	136	81	15	425	120	Good	67	10	Yes	Yes
8	6.54	132	110	0	108	124	Medium	76	10	No	No
9	4.69	132	113	0	131	124	Medium	76	17	No	Yes

اینکار مانند شکل بالا انجام شده است.

۲.۳)پیش پردازش دادگان

یافتن داده های ناقص یا گمشده:

2.3- Pre Processing

```
#Checking for missing values
company_data.isnull().sum()
```

	0
Sales	0
CompPrice	0
Income	0
Advertising	0
Population	0
Price	0
ShelveLoc	0
Age	0
Education	0
Urban	0
US	0

dtype: int64

```
[51] # Removing rows with missing values
     company_data = company_data.dropna()
```

وجود داده‌های تکراری در دادگان می‌تواند در روند یادگیری ماشـــین ایجاد ســـوگیری(بایاس) کند و در نتیجه نهایی و کارکرد ماشین خطا ایجاد کند. تشخیص و حذف آن‌ها به صورت زیر است:

```
[52] num_duplicates = company_data.duplicated().sum()
     company_data = company_data.drop_duplicates()
     print(f"Number Of Duplicate Rows:{num_duplicates}")

Number Of Duplicate Rows:0
```

تبدیل ویژگی‌های دسته‌ای به عددی نیز به صورت زیر انجام شده است:

```
# Encoding
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder

label_encoder = LabelEncoder()
# "One-Hot Encoding" Used for "Urban" and "US". since they dont have an inherent order (nominal features)
#using Label Encoding can mislead the model into assuming an order where there isn't one. This can
#potentially affect model performance negatively.
company_data = pd.get_dummies(company_data, columns=['Urban', 'US'], drop_first=True)

#"LabelEncoder" can be Used for Shelveloc since it can be efficient and work well for features with an inherent order (natural ranking)
# company_data['ShelveLoc'] = label_encoder.fit_transform(company_data['ShelveLoc'])
# but This code by default orders the classes alphabetically, not by any intrinsic "goodness", Thus:
#*** Models like Decision Trees or Random Forests don't really care much about numerical order – they just split based on feature values.
#But models like Linear Regression, Logistic Regression, SVM, or anything that assumes numeric relationships will think "Good" (1) is between
#Another Fix:
#order_map = {'Bad': 0, 'Medium': 1, 'Good': 2}
#company_data['ShelveLoc'] = company_data['ShelveLoc'].map(order_map)


#Ordinal Encoder Is
ord_enc = OrdinalEncoder(categories=[['Bad', 'Medium', 'Good']])
company_data[['ShelveLoc']] = ord_enc.fit_transform(company_data[['ShelveLoc']])
#or
#company_data[['ShelveLoc']] = ord_enc.fit_transform(company_data[['ShelveLoc']]).astype(int)  for integer instead of float
```

```
# Just Making Sure Urban_YES and US_YES are int (so that we dont run into errors during training)
company_data[['Urban_Yes', 'US_Yes']] = company_data[['Urban_Yes', 'US_Yes']].astype(int)
company_data.dtypes
```

	0
Sales	float64
CompPrice	int64
Income	int64
Advertising	int64
Population	int64
Price	int64
ShelveLoc	float64
Age	int64
Education	int64
Urban_Yes	int64
US_Yes	int64

dtype: object

```
[56] company_data
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_Yes	US_Yes
0	9.50	138	73	11	276	120	0.0	42	17	1	1
1	11.22	111	48	16	260	83	2.0	65	10	1	1
2	10.06	113	35	10	269	80	1.0	59	12	1	1
3	7.40	117	100	4	466	97	1.0	55	14	1	1

همچنین تبدیل ویژگی دسته‌ای sales به عددی به کمک معیار میانگین و به دو دسته انجام شده است زیرا تعادل میان دو دسته را به خوبی رعایت میکند(مقایسه با دو روش دیگر Kmeans , quantile در کد انجام شده است):

```
[ ] #Using Mean as the Threshold
    company_data['Sales'].mean(),(company_data['Sales'] > company_data['Sales'].mean()).sum(),(company_data['Sales'] <= company_data['Sales'].mean

(np.float64(7.496325000000001), np.int64(199), np.int64(201))

    #Using Kmean Clustering as Threshold
    from sklearn.cluster import KMeans
    kmeans = KMeans(n_clusters=3, random_state=52)
    new_df = pd.DataFrame(kmeans.fit_predict(company_data[['Sales']]))
    print(kmeans.cluster_centers_),new_df.value_counts()

[[11.6265    ]
 [ 7.97235955]
 [ 4.57274648]]
(None,
 0
 1    178
 2    142
 0     80
 Name: count, dtype: int64)

[ ] # Calculate thresholds using quantiles
    thresholds = np.quantile(company_data['Sales'], [0.33, 0.67])
    thresholds,company_data['Sales'].nlargest(5)
    #8.68 is far from 11.62 which is the 3rd Mean cluster value and is not a good representetive of outliers such as 16.27

(array([6.03  , 8.6833]),
 376    16.27
 316    15.63
 25     14.90
 367    14.37
 18     13.91
 Name: Sales, dtype: float64)
```

```
# Based On the 3 methods used Above, Mean is the best threshold since the Classes would be More balanced
company_data['Sales_Categorical'] = pd.cut(company_data['Sales'],
                                           bins=[-np.inf, company_data['Sales'].mean(), np.inf],
                                           labels=['Low', 'High'])

#Reorginizing DataFrame for Better Look

#Removing Sales column and replacing it with "Sales_Category"
company_data = company_data.drop('Sales', axis=1)

#company_data = company_data[['Sales_Categorical'] + [col for col in company_data.columns if col != 'Sales_Categorical']] *** or the following:
company_data = pd.concat([company_data[['Sales_Categorical']], company_data[[col for col in company_data.columns if col != 'Sales_Categorical']]], axis=1)


company_data
```

	Sales_Categorical	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_Yes	US_Yes
0	High	138	73	11	276	120	0.0	42	17	1	1
1	High	111	48	16	260	83	2.0	65	10	1	1
2	High	113	35	10	269	80	1.0	59	12	1	1
3	Low	117	100	4	466	97	1.0	55	14	1	1
4	Low	141	64	3	340	128	0.0	38	13	1	0
...	...	...	...	...	...	...	...	...	...	...	...
395	High	138	108	17	203	128	2.0	33	14	1	1
396	Low	139	23	3	37	120	1.0	55	11	0	1
397	Low	162	26	12	368	159	1.0	40	18	1	1
398	Low	100	79	7	284	95	0.0	50	12	1	1
399	High	134	37	0	27	120	2.0	49	16	1	1

400 rows × 11 columns

```
company_data = pd.get_dummies(company_data, columns=['Sales_Categorical'], drop_first=True).astype(int)
company_data
```

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_Yes	US_Yes	Sales_Categorical_High
0	138	73	11	276	120	0	42	17	1	1	1
1	111	48	16	260	83	2	65	10	1	1	1
2	113	35	10	269	80	1	59	12	1	1	1
3	117	100	4	466	97	1	55	14	1	1	0
4	141	64	3	340	128	0	38	13	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...
395	138	108	17	203	128	2	33	14	1	1	1
396	139	23	3	37	120	1	55	11	0	1	0
397	162	26	12	368	159	1	40	18	1	1	0
398	100	79	7	284	95	0	50	12	1	1	0
399	134	37	0	27	120	2	49	16	1	1	1

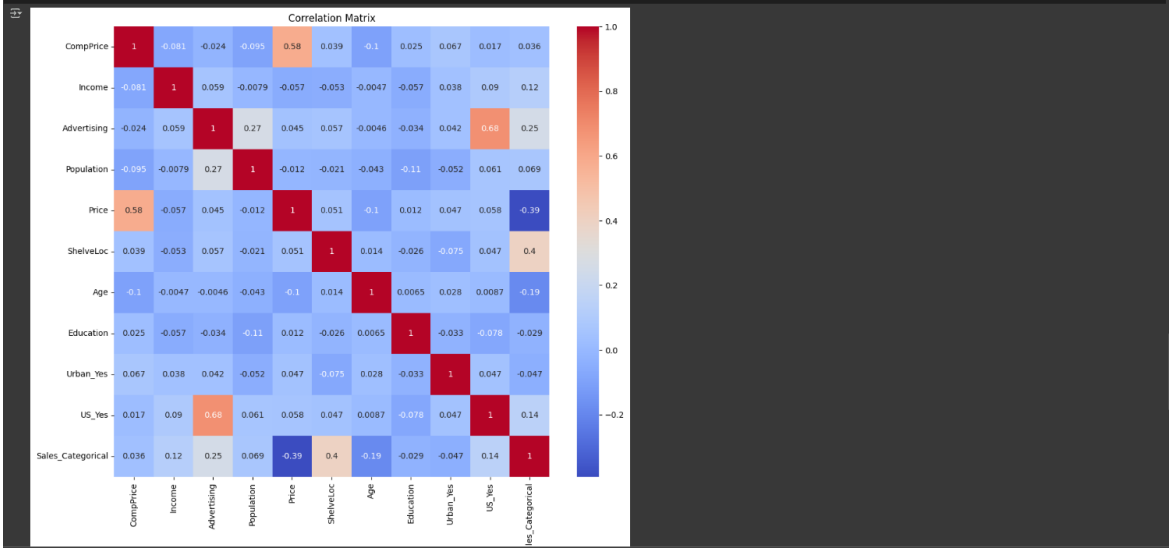
400 rows × 11 columns

```
company_data["Sales_Categorical"]=company_data["Sales_Categorical_High"]
company_data = company_data.drop(['Sales_Categorical_High'], axis=1)
company_data
```

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_Yes	US_Yes	Sales_Categorical
0	138	73	11	276	120	0	42	17	1	1	1
1	111	48	16	260	83	2	65	10	1	1	1
2	113	35	10	269	80	1	59	12	1	1	1

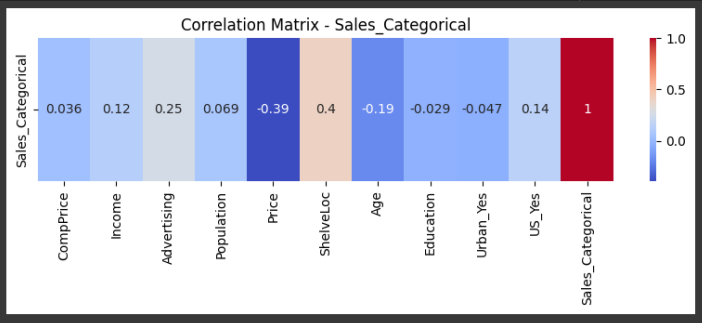
ترسیم ماتریس همبستگی نیز به صورت زیر است:

```
correlation_matrix = company_data.corr()
# ترسیم heatmap
plt.figure(figsize=(12, 10))  # تنظیم اندازه شکل
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



همچنین برای یافتن بیشترین همبستگی داده ها با sales_categorical با فرض آستانه ۰.۲، رسم و اسم ویژگی ها به صورت زیر است:

```
plt.figure(figsize=(10, 2))
sns.heatmap(correlation_matrix[['Sales_Categorical']].T, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix - Sales_Categorical')
plt.show()
#.T is used to show plot horizontally
```




```
target='Sales_Categorical'
threshold=0.2
Highcors = correlation_matrix[target][abs(correlation_matrix[target]) > threshold].index.tolist() #Choosing Features with Corr>0.2
Highcors.remove('Sales_Categorical')
Highcors
```

```
['Advertising', 'Price', 'ShelveLoc']
```

۳.۳)محاسبه آنتروپی داده‌ها

با توجه به فرمول داده شده، محاسبه آنتروپی داده‌ها به کمک calculate_entropy به صورت زیر است:

```
3.3-Entropy Calculation
```

```
def calculate_entropy(labels):
    p=labels.value_counts()/len(labels)
    return -sum(p*np.log2(p))
```

```
calculate_entropy(company_data['Sales_Categorical'])
```

```
0.9999819662368479
```

۴.۳)محاسبه بهره اطلاعات (information gain)

محاسبه بهره‌اطلاعات نیز اینگونه است‌که ابتدا آنتروپی والد برای ویژگی مورد نظر محاسبه شده و با فرض مقدار اولیه صفر برای فرزند، آنتروپی زیرمجموعه ها و وزن‌ها محاسبه و از آنتروپی والد کاسته می‌شوند:

```
4.3-Information Gain
```

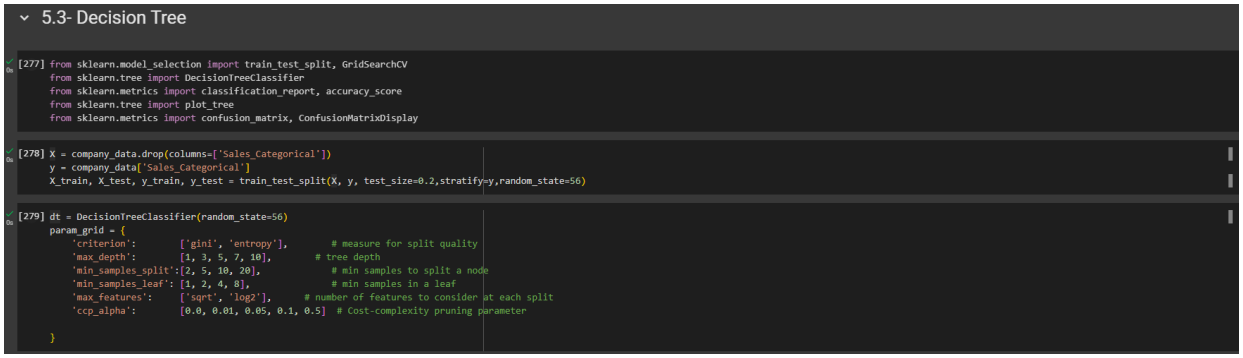
```
def info_gain(data, child, parent):
    # Entropy of parent
    entropy_parent = calculate_entropy(data[parent])

    # Entropy of child
    entropy_child = 0
    for value in data[child].unique():
        subset = data[data[child] == value]
        wi = len(subset) / len(data)
        entropy_child += wi * calculate_entropy(subset[parent])

    return entropy_parent - entropy_child
```

```
info_gain(company_data, 'ShelveLoc', 'Sales_Categorical')
```

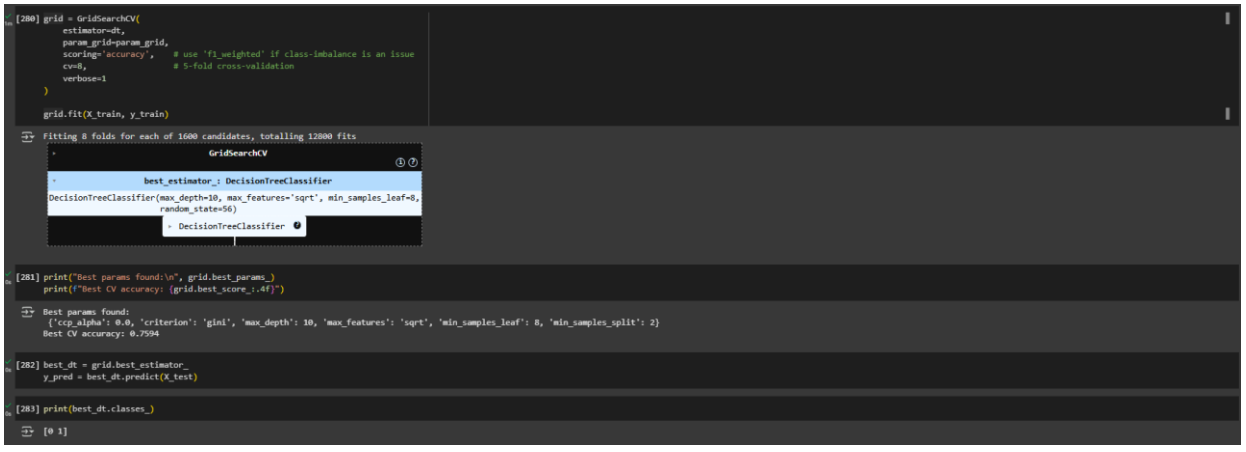
```
0.12802289274162826
```

۵.۳)درخت تصمیم:

- درخت تصمیم در صورتی که داده‌های آموزش بیش از حد زیاد شوند، می‌تواند دچار بیش برازش شود که موجب خطا بر روی داده‌های تست شده و عملکرد نامناسبی دارد.هرس کردن(pruning) روشی برای ساده سازی و جلوگیری از این اشتباه با حذف شاخه‌هایی است که ممکن است اطلاعات جدیدی به مدل اضافه نکنند. از معایب آن نیز می‌توان به احتمال کم‌برازشی و نیاز به انتخاب پارامترهای بیشتر و درست (هزینه محاسباتی بالاتر) اشاره نمود.
- GridSearchCV ابزاری در کتابخانه Scikit-learn است که برای یافتن بهترین ترکیب از پارامترهای مدل استفاده می‌شود. این ابزار، مجموعه‌ای از مقادیر ممکن برای هر پارامتر را دریافت کرده و با استفاده از اعتبارسنجی متقابل، تمام ترکیب‌های ممکن را آزمایش می‌کند و به شکل exhaustive عمل میکند. سپس با محاسبه دقت یا معیارهای دیگر برای هر ترکیب، بهترین تنظیمات را که عملکرد مدل را بیشینه می‌کنند انتخاب می‌نماید. به این صورت، GridSearchCV به انتخاب خودکار و بهینه پارامترها کمک می‌کند و از بیش برازش نیز جلوگیری می‌کند.

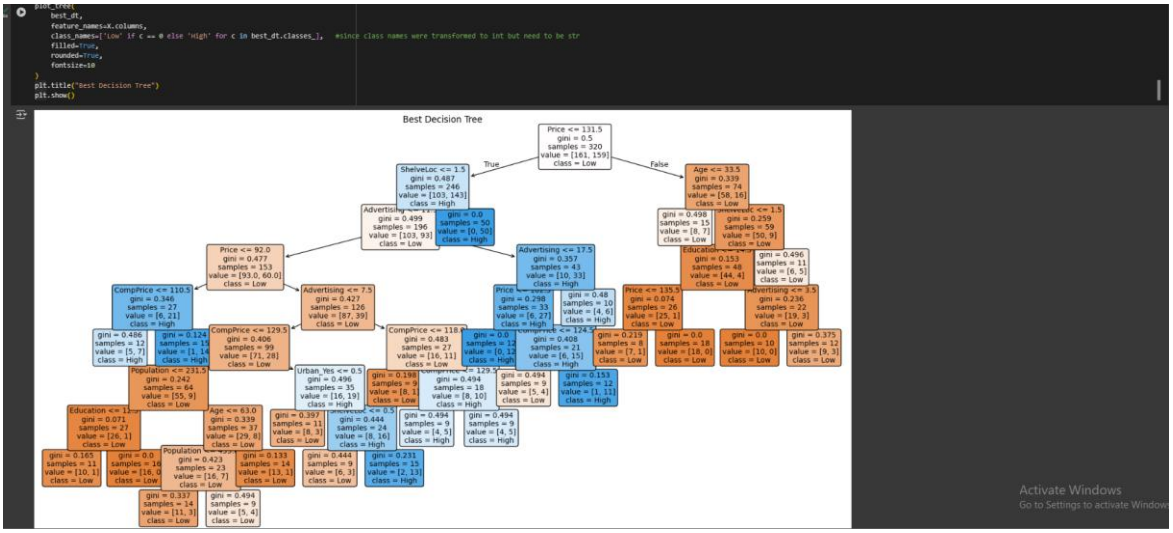


```
5.3- Decision Tree

[277] from sklearn.model_selection import train_test_split, GridSearchCV
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import classification_report, accuracy_score
      from sklearn.tree import plot_tree
      from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

[278] X = company_data.drop(columns=['Sales_Categorical'])
      y = company_data['Sales_Categorical']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,stratify=y,random_state=56)

[279] dt = DecisionTreeClassifier(random_state=56)
      param_grid = {
          'criterion':       ['gini', 'entropy'],       # measure for split quality
          'max_depth':       [1, 3, 5, 7, 10],        # tree depth
          'min_samples_split':[2, 5, 10, 20],           # min samples to split a node
          'min_samples_leaf': [1, 2, 4, 8],             # min samples in a leaf
          'max_features':    ['sqrt', 'log2'],     # number of features to consider at each split
          'ccp_alpha':       [0.0, 0.01, 0.05, 0.1, 0.5]  # Cost-complexity pruning parameter

      }
```



```
[280] grid = GridSearchCV(
          estimator=dt,
          param_grid=param_grid,
          scoring='accuracy',    # use 'f1_weighted' if class-imbalance is an issue
          cv=8,                  # 5-fold cross-validation
          verbose=1
      )

      grid.fit(X_train, y_train)

Fitting 8 folds for each of 1600 candidates, totalling 12800 fits
GridSearchCV
best_estimator_: DecisionTreeClassifier
DecisionTreeClassifier(max_depth=10, max_features='sqrt', min_samples_leaf=8,
                       random_state=56)
DecisionTreeClassifier

[281] print("Best params found:\n", grid.best_params_)
      print(f"Best CV accuracy: {grid.best_score_:.4f}")

Best params found:
 {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 8, 'min_samples_split': 2}
Best CV accuracy: 0.7594

[282] best_dt = grid.best_estimator_
      y_pred = best_dt.predict(X_test)

[283] print(best_dt.classes_)

['0' '1']
```

- رسم درخت تصمیم نهایی نیز بصورت زیر است:



- برای بررسی بیش برازش یا کم برازش باید مدل را هم روی داده های تست و هم آموزش بررسی نمود؛ اگر دقت مدل روی داده های آموزش بسیار بالا ولی روی داده‌های تست بسیار پایین باشد، بیش برازش رخ داده است و اگر دقت مدل هم روی داده‌های آموزش و هم آزمون پایین باشد، کم برازشی رخ داده است. به طور کلی بیش برازش زمانی رخ می‌دهد که درخت بسیار عمیق باشد، معیار جداسازی بدون کنترل مناسب بوده و داده‌ها نویز بالایی داشته و یا دارای ویژگی‌های نامربوطی باشند. کم برازشی نیز در صورتی رخ می‌دهد که عمق درخت کم بوده، حداقل تعداد نمونه برای جداسازی زیاد باشد و اطلاعات کافی از داده ها برای ساخت مدل استخراج نشده باشد. از راهکارهای مناسب برای جلوگیری نیز استفاده از درخت‌های مجموعه‌ای مانند Random Forest، هرس کردن، و انتخاب عمق و مقادیر جداسازی مناسب است.

```
[288] #Checking for overfitting and underfitting

    # Predict on training and test sets
    y_train_pred = best_dt.predict(X_train)
    y_test_pred = best_dt.predict(X_test)

    # Calculate accuracies
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)

    print(f"Training Accuracy: {train_accuracy:.3f}")
    print(f"Test Accuracy:     {test_accuracy:.3f}")

    # Simple diagnostic
    if train_accuracy > 0.95 and (train_accuracy - test_accuracy) > 0.1:
        print("\n Possible Overfitting: Model performs very well on training data but poorly on test data.")
    elif train_accuracy < 0.7 and test_accuracy < 0.7:
        print("\n Possible Underfitting: Model does not perform well on either training or test data.")
    else:
        print("\n Model seems to be generalizing well.")

Training Accuracy: 0.822
Test Accuracy:     0.738

 Model seems to be generalizing well.
```

- خروجی معیارهای ارزیابی برای داده های آزمون به صورت زیر است:

```
print("\nTest set performance:")
print(classification_report(y_test, y_pred, target_names=['Low' if c == 0 else 'High' for c in best_dt.classes_]))  #its possible not to write target_names
print("Test accuracy: ", accuracy_score(y_test, y_pred))
```

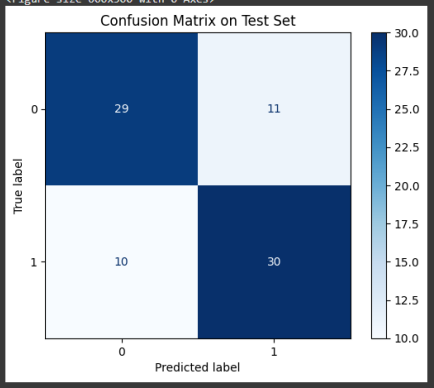
```
Test set performance:
              precision    recall  f1-score   support

         Low       0.74      0.72      0.73        40
        High       0.73      0.75      0.74        40

    accuracy                           0.74        80
   macro avg       0.74      0.74      0.74        80
weighted avg       0.74      0.74      0.74        80

Test accuracy:  0.7375
```

همانگونه که مشاهده می‌شود، مدل با دقت کلی ۷۳/۷۵٪ عملکرد قابل قبولی روی داده‌های آزمون دارد و مقادیر recall,precision و F1-score نیز برای هردو کلاس نزدیک می‌باشند که نشان دهنده عدم وجود سوگیری می‌باشد(در ابتدا مقدار میانگین برای جداسازی فروش استفاده شد تا از سوگیری جلوگیری شود).

- ماتریس درهم ریختگی برای داده‌های آزمون نیز به صورت زیر است(برچسب‌ها: ۰ کم و ۱ زیاد):

```
cm = confusion_matrix(y_test, y_pred, labels=best_dt.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=best_dt.classes_)

plt.figure(figsize=(6, 5))
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix on Test Set")
plt.show()
```

```
<Figure size 600x500 with 0 Axes>
```



همانگونه که مشاهده می‌شود، ماتریس درهم ریختگی نتایج را به خوبی نشان می‌دهد چرا که مقادیر ۰ واقعی درست تخمین زده شده ۲۹ و ۱ واقعی درست تخمین زده شده ۳۰ تا می‌باشند که بسیار بیشتر از مقادیر اشتباه می‌باشند.

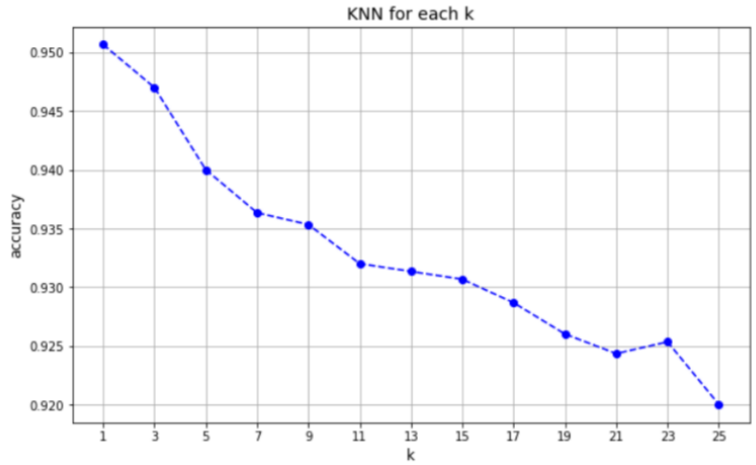
۲.پرسش دو-MNIST

آ) روش StandardScaler داده ها را به گونه ای تبدیل میکند که میانگین صفر و انحراف معیار یک باشد اما روش MinMaxScaler این روش دادهها را به بازهی[0,1] یا [-1,1] تبدیل می کند.

هر پیکسل در MNIST یک مقدار بین 0 تا 255 دارد. MinMaxScaler به سادگی این مقادیر را به بازهی [0,1] تبدیل میکند، این تبدیل خطی است و ساختار اصلی داده ها را حفظ می کند و برای پردازش تصاویر بسیار مناسب است.

ج) د)

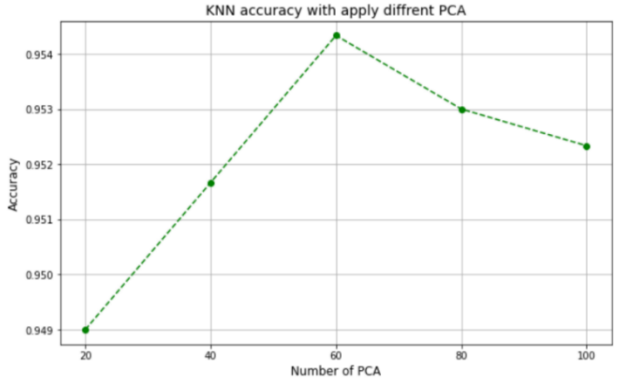
```
best value for k: 1 accuracy 0.9507
```




```
k = 1: accuracy = 0.9507
k = 3: accuracy = 0.9470
k = 5: accuracy = 0.9400
k = 7: accuracy = 0.9363
k = 9: accuracy = 0.9353
k = 11: accuracy = 0.9320
k = 13: accuracy = 0.9313
k = 15: accuracy = 0.9307
k = 17: accuracy = 0.9287
k = 19: accuracy = 0.9260
k = 21: accuracy = 0.9243
k = 23: accuracy = 0.9253
k = 25: accuracy = 0.9200
```

د) در روش PCA با حفظ واریانس اطلاعاتی، ابعاد داده را کاهش میدهد. با افزایش تعداد مؤلفه های PCA، دقت بهبود می یابد و ام بعد از مقداری (در اینجا ۶۰) علاوه بر این که تاثیری ندارد حتی باعث کاهش دقت نیز می شود.

```
Component: 20 | accuracy: 0.9490
Component: 40 | accuracy: 0.9517
Component: 60 | accuracy: 0.9543
Component: 80 | accuracy: 0.9530
Component: 100 | accuracy: 0.9523
```



1.پرسش یک-پیامک اسپم

الف)

طبقه‌بندی Naive Bayes:

این روش بر اساس قضیه بیز کار می‌کند و فرض می‌کند که ویژگی ها مستقل از یکدیگر هستند. با وجود سادگی، در بسیاری از موارد عملکرد خوبی دارد، به ویژه در پردازش متن و داده های با ابعاد بالا. و البته محاسبات آن سریع و کم هزینه است.

طبقه‌بندی Optimal Bayes:

این روش بهترین طبقه بند ممکن را ارائه می‌دهد، به شرطی که توزیع داده ها به‌طور کامل شناخته شده باشد. برخلاف Naive Bayes، هیچ فرض ساده‌کننده‌ای ندارد و وابستگی بین ویژگی‌ها را در نظر می‌گیرد. در عمل، به دلیل نیاز به اطلاعات کامل از توزیع داده ها، پیاده‌سازی آن دشوار است.

در کل می توان گفت؛ Naive Bayes یک روش عملی و سریع با فرض ساده‌کننده است، در حالی که Optimal Bayes یک روش تئوری و ایده‌آل است که در عمل به ندرت قابل استفاده است.

ب)

برای مجموعه داده پیامک های اسپم، Multinomial Naive Bayes بنظر مناسب تر است چرا که؛

مجموعه داده پیامک ها متنی هستند و با روش‌هایی مانند Bag of Words به اعداد تبدیل می‌شوند. این تبدیل معمولاً مبتنی بر تعداد تکرار کلمات است مثلاً کلمه **جایزه** ۲ بار در پیامک آمده. مدل Multinomial به‌طور خاص برای داده‌های شمارشی (مانند تعداد دفعات تکرار یک کلمه) طراحی شده و در طبقه‌بندی متن عملکرد بهتری دارد. مدل Bernoulli فقط وجود یا عدم وجود کلمات را بررسی می‌کند و اطلاعات تکرار را نادیده می‌گیرد. مدل Gaussian برای داده‌های پیوسته با توزیع نرمال مناسب است و با داده های متنی سازگاری / ندارد.

برای مثال در تشخیص اسپم، تکرار برخی کلمات مثل **مسابقه** و **رایگان** اغلب نشانه از احتمال اسپم بودن است. Multinomial این اطلاعات را به خوبی مدل می‌کند.

ج)

```
Accuracy of Model is: 98.21%
- Confusion Matrix:
  TP: 131 | FP: 6
  FN: 14 | TN: 964
- Accuracy: 98.21%
- Precision: 95.62%
- Recall: 90.34%
- F1 score: 92.91%
```

د) Bernoulli Naive Bayes عملکرد بهتری دارد چون برای داده‌های متنی با ویژگی‌های دودویی (وجود یا عدم وجود کلمه) مناسب است در حالی که Gaussian Naive Bayes به دلیل فرض توزیع نرمال، برای داده‌های متنی (که معمولاً پراکنده و غیرنرمال هستند) ضعیف عمل می‌کند.

```
Bernoulli Naive Bayes:
Confusion matrix:
[[1451    2]
 [  34  185]]
Accuracy: 97.85%
Precision: 98.93%
Recall: 84.47%
```

```
Gaussian Naive Bayes:
confusion matrix:
[[1311  142]
 [  18  201]]
Accuracy: 90.43%
Precision: 58.60%
Recall: 91.78%
```

با توجه به نتایج مشخص است که Multinomial در accuracy و recall عملکرد بهتری دارد لذا اگر صرفا هدف دقت باشد مدل اول بهترین است اما اگر هدف کاهش False Alarm باشد باید از Bernoulli و اگر هدف کاهش miss detection باشد Gaussian بهتر است.