به نام خدا

گزارش تمرین سوم که دیتاست اطلاعات مشخصات و قیمت گوشی همراه می باشد.

محمدسعید حیدری 400422075

در این گزارش اقداما تی که در جریان حل تسکهای مشخص شده برای تمرین 3 انجام داده شده و توضیحات و تحلیلهای مربوط به آنها ارائه شده است.

لازم به ذکر است ترتیب تسک ها را در این تمرین جابجا کرده ام پس فقط طبق فایل pdf موجود این تمرین بررسی شود

جواب سوال ۱:

پیاده سازی روش forward feature selection با استفاده از AUC :

```
def forward_selection_with_auc(x, y, test_size=0.2):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=79)
    initial_features = x.columns.tolist()
    best_features = []

    while len(initial_features)>0:
        print("ok")
        auc_value = {}
        remaining_features = list(set(initial_features) - set(best_features))
        new_x_train = x_train[remaining_features]
        new_x_test = x_test[remaining_features]

        if len(best_features) == 0:
            for new_column in remaining_features:
                model = LogisticRegression().fit(new_x_train[new_column].to_numpy().reshape(-1, 1), y_train)
                y_predict = model.predict(new_x_test[new_column].to_numpy().reshape(-1, 1))
                auc_value[new_column] = auc_from_scratch(y_test, y_predict, 100)

            max_auc_value = max(auc_value)

            print(auc_value[max_auc_value])

            best_features.append(max_auc_value)

        else :
            best_x_train = x_train[best_features]
            best_x_test = x_test[best_features]

            for new_column in remaining_features:
                model = LogisticRegression().fit(best_x_train.join(new_x_train[new_column]), y_train)
                y_predict = model.predict(best_x_test.join(new_x_test[new_column]))
                auc_value[new_column] = auc_from_scratch(y_test, y_predict, 100)

            max_auc_value = max(auc_value)

            model = LogisticRegression().fit(best_x_train, y_train)
            y_predict = model.predict(best_x_test)
            best_auc_value = auc_from_scratch(y_test, y_predict, 100)

            print(auc_value[max_auc_value])
            print(best_auc_value)
            best_features.append(max_auc_value)
            if(best_auc_value<auc_value[max_auc_value]):
                best_features.append(max_auc_value)
            else:
                break

    return best_features
```

در این روش ما از یک لیست تهی از فیچر ها شروع کرده و با استفاده از مدل لجستیک و یادگیری آن مرحله به مرحله هر فیچری که AUC ما را افزایش دهد به لیست اضافه میکنیم.

```
1 forward_selection_with_auc(x, binary_y)
```

```
ok
1.0
ok
0.48958333333333337
1.0

['wifi', 'touch_screen']
```

در روش دیگری از forward selection با استفاده از P-value فیچر هایی که P-value بالاتری برخوردار باشند به لیست ما اضافه می شوند نکته ای که باید مد نظر داشته باشیم این است که در اینجا حد آستانه ما به نوعی ضریب اطمینانی است که در ابتدا تعیین میکنیم اگر مقدار احتمال ما از این مقدار کمتر باشد لیست نهایی میشود.

```python
def forward_selection(x, y, significance_level=0.05):
    initial_features = x.columns.tolist()
    best_features = []
    while (len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features, dtype=np.float64)
        for new_column in remaining_features:
            model = sm.OLS(y, sm.add_constant(x[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value<significance_level):
            best_features.append(new_pval.idxmin())
        else:
            break
    return best_features
```

```python
forward_selection(x, binary_y)
```

```
['ram', 'battery_power', 'px_width', 'px_height', 'n_cores']
```

جواب سوال ۲:

AUC

```python
new_x = df[['wifi', 'touch_screen']]
# train test split dataset
x_train, x_test, y_train, y_test = train_test_split(new_x, binary_y, test_size=0.2, random_state=79)
```

```python
# Create model
model = LogisticRegression()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.515
```

```python
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.49      0.49      0.49       192
           1       0.53      0.54      0.54       208

    accuracy                           0.52       400
   macro avg       0.51      0.51      0.51       400
weighted avg       0.51      0.52      0.51       400
```

P-value

```python
new_x = df[['ram', 'battery_power', 'px_width', 'px_height', 'n_cores']]
# train test split dataset
x_train, x_test, y_train, y_test = train_test_split(new_x, binary_y, test_size=0.2, random_state=79)
```

```python
# Create model
model = LogisticRegression()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9925
```

```python
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       192
           1       1.00      0.99      0.99       208

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

جواب سوال ۳ و ۴ :

AUC

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

```
principalDf.head()
```

	principal component 1	principal component 2
0	430.597094	-795.788231
1	504.984735	696.622368
2	473.329828	763.942136
3	639.822324	779.691180
4	-718.985184	382.304525

```
# train test split dataset
x_train, x_test, y_train, y_test = train_test_split(principalDf, binary_y, test_size=0.2, random_state=79)
```

```
# Create model
model = LogisticRegression()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.94
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       192
           1       0.95      0.93      0.94       208

    accuracy                           0.94       400
   macro avg       0.94      0.94      0.94       400
weighted avg       0.94      0.94      0.94       400
```

P-value

```
pca = PCA(n_components=5)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['pc1', 'pc2', 'pc3', 'pc4', 'pc5'])
```

```
# train test split dataset
x_train, x_test, y_train, y_test = train_test_split(principalDf, binary_y, test_size=0.2, random_state=79)
```

```
# Create model
model = LogisticRegression()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9875
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       192
           1       0.98      1.00      0.99       208

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

جواب سوال ۵ و ۶ :

الف)

Binning

- Three different types of binning

Type1

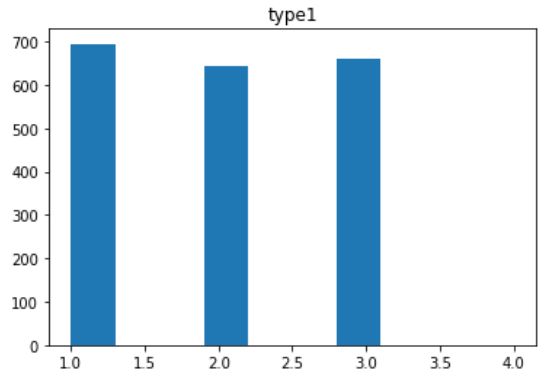
```
x = df.drop(["price_range"], axis=1)
x["battery_power"].value_counts(bins=3)
```

```
(499.502, 1000.0]    697
(1499.0, 1998.0]     662
(1000.0, 1499.0]     641
Name: battery_power, dtype: int64
```

```
bins_1 = np.array([0, 1000.0, 1499.0, 2000])
digitized_1 = np.digitize(x["battery_power"], bins_1)
```

```
x["battery_power"] = digitized_1
x["battery_power"].value_counts()
```

```
plt.hist(digitized_1)
plt.title("type1")
plt.show()
```




```
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
# Create model
model = LogisticRegression(solver='newton-cg', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9825
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       192
           1       0.98      0.99      0.98       208

    accuracy                           0.98       400
   macro avg       0.98      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

Type2

```
x = df.drop(["price_range"], axis=1)
x["battery_power"].value_counts(bins=4)
```

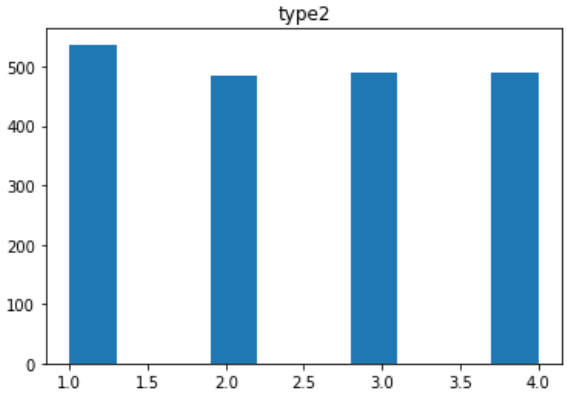
```
(499.502, 875.25]    537
(1623.75, 1998.0]    490
(1249.5, 1623.75]    489
(875.25, 1249.5]     484
Name: battery_power, dtype: int64
```

```
bins_2 = np.array([0, 875.25, 1249.5, 1623.75, 2000])
digitized_2 = np.digitize(x["battery_power"], bins_2)
```

```
x["battery_power"] = digitized_2
x["battery_power"].value_counts()
```

```
1    537
4    490
3    489
2    484
Name: battery_power, dtype: int64
```

```
plt.hist(digitized_2)
plt.title("type2")
plt.show()
```




```
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
# Create model
model = LogisticRegression(solver='newton-cg', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.985
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       192
           1       0.99      0.99      0.99       208

    accuracy                           0.98       400
   macro avg       0.98      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

Type3

```
x = df.drop(["price_range"], axis=1)
x["battery_power"].value_counts(bins=5)
```

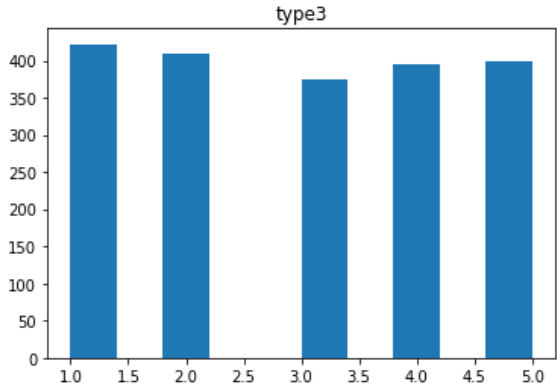
```
(499.502, 800.4]    422
(800.4, 1099.8]     409
(1698.6, 1998.0]    400
(1399.2, 1698.6]    394
(1099.8, 1399.2]    375
Name: battery_power, dtype: int64
```

```
bins_3 = np.array([0, 800.4, 1099.8, 1399.2, 1698.6, 2000])
digitized_3 = np.digitize(x["battery_power"], bins_3)
```

```
x["battery_power"] = digitized_3
x["battery_power"].value_counts()
```

```
1    422
2    409
5    400
4    394
3    375
Name: battery_power, dtype: int64
```

```
plt.hist(digitized_3)
plt.title("type3")
plt.show()
```




```
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
# Create model
model = LogisticRegression(solver='newton-cg', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

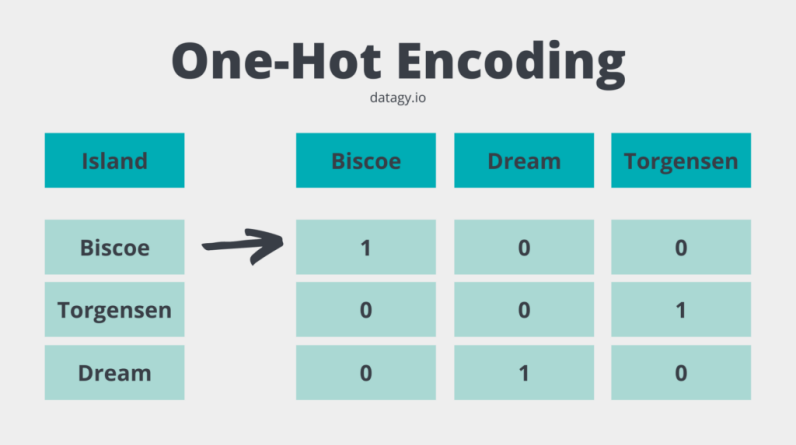
```
0.9875
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       192
           1       0.98      1.00      0.99       208

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

ب)

One hot encoding یکی از روش‌های تبدیل داده‌ها برای آماده‌سازی آن برای الگوریتم و پیش‌بینی بهتر است. با one-hot، هر مقدار طبقه بندی را به یک ستون دسته بندی جدید تبدیل می کنیم و مقدار باینری 1 یا 0 را به آن ستون ها اختصاص می دهیم. هر عدد صحیح به صورت یک بردار باینری نمایش داده می شود.




```
new_x = df.drop(["price_range"], axis=1)
```

```
# finding categorical features (binary values can be considered as categorical features!)
df.columns
```

```
Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi', 'price_range'],
      dtype='object')
```

```
df[['blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi']].head()
```

	blue	dual_sim	four_g	three_g	touch_screen	wifi
0	0	0	0	0	0	1
1	1	1	1	1	1	0
2	1	1	1	1	1	0
3	1	0	0	1	0	0
4	1	0	1	1	1	0

```
# Turn the categories into numbers
categorcal_features = ['blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi']
one_hot = OneHotEncoder()
transformer = ColumnTransformer([("one_hot",
                                  one_hot,
                                  categorcal_features)],
                                  remainder="passthrough")

onehot_encoding_x = transformer.fit_transform(new_x)
onehot_encoding_x
```

```
array([[ 1.,  0.,  1., ...,  9.,  7., 19.],
       [ 0.,  1.,  0., ..., 17.,  3.,  7.],
       [ 0.,  1.,  0., ..., 11.,  2.,  9.],
       ...,
       [ 1.,  0.,  0., ...,  9.,  1.,  5.],
       [ 1.,  0.,  1., ..., 18., 10., 19.],
       [ 0.,  1.,  0., ..., 19.,  4.,  2.]])
```

```
onehot_encoding_x.shape
```

```
(2000, 26)
```

```
x_train, x_test, y_train, y_test = train_test_split(onehot_encoding_x, binary_y, test_size=0.2, random_state=79)
```

```
# Create model
model = LogisticRegression(solver='newton-cg', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9875
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       192
           1       0.99      0.99      0.99       208

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

One hot encoding داده های آموزشی ما را مفیدتر و گویاتر می کند و می توان آن را به راحتی تغییر مقیاس داد. با استفاده از مقادیر عددی، ما به راحتی احتمالی را برای مقادیر خود تعیین می کنیم. به ویژه، One hot encoding برای مقادیر خروجی ما استفاده می شود، زیرا پیش بینی های ظریف تری نسبت به برچسب های منفرد ارائه می دهد.

ج)

چولگی و کورتوز

به طور خلاصه، چولگی معیاری برای سنجش تقارن است. به طور دقیق، این معیاری برای عدم تقارن است. این بدان معنی است که هر چه عدد بزرگتر باشد، داده های شما فاقد تقارن هستند (یعنی عادی نیست). از سوی دیگر، Kurtosis معیاری است که نشان می‌دهد داده‌های شما نسبت به توزیع نرمال، سنگین یا سبک هستند.

برای تعریف ریاضی بیشتر هر دو معیار اینجا را ببینید. یک راه خوب برای بررسی بصری داده ها از نظر چولگی یا کشیدگی، استفاده از هیستوگرام است. البته توجه داشته باشید که البته تست های آماری مختلفی نیز وجود دارد که می توان از آنها برای آزمایش اینکه آیا داده های شما به طور معمول توزیع شده است استفاده کرد.

- چگونه داده های چوله را در پایتون مدیریت می کنید؟

یکی از روش‌های مدیریت داده‌های چوله به سمت راست یا چپ، انجام تبدیل لگاریتمی روی داده‌های ما است. به عنوان مثال، np.log(x) متغیر x را در پایتون تبدیل لگاریتمی می کند. گزینه های دیگری و همچنین تبدیل Box-Cox و Square-Root وجود دارد.

Square-Root

روش ریشه دوم معمولاً زمانی استفاده می شود که داده های شما دارای انحراف متوسط باشد. اکنون استفاده از ریشه مربع (به عنوان مثال (sqrt(x) تبدیلی است که تأثیر متوسطی بر شکل توزیع دارد. به طور کلی برای کاهش داده های چولگی سمت راست استفاده می شود. در نهایت، جذر را می توان روی مقادیر صفر اعمال کرد و بیشتر در داده های شمارش شده استفاده می شود.

Log transformation

تبدیل لگاریتمی یک تبدیل قوی است که تأثیر عمده ای بر شکل توزیع دارد. این تکنیک، مشابه روش ریشه مربع، اغلب برای کاهش چولگی سمت راست استفاده می شود. با این حال، شایان ذکر است که نمی توان آن را برای مقادیر صفر یا منفی اعمال کرد.

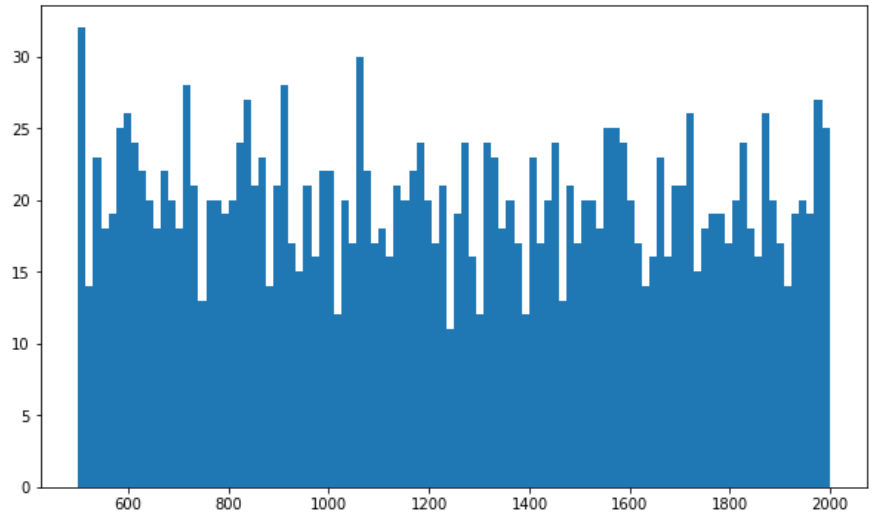
Box-Cox transformation

همان طور که احتمالاً متوجه شده اید، تبدیل باکس-کاکس هم تکنیکی برای تبدیل داده های غیر عادی به شکل عادی. این یک روش برای شناسایی یک توان مناسب (Lambda = l) برای استفاده برای تبدیل داده های کج است.

در حال حاضر، تکنیک های تبدیل ذکر شده در بالا متداول ترین مورد استفاده هستند. با این حال، بسیاری از روش های دیگر نیز وجود دارد، که می تواند برای تبدیل متغیرهای وابسته اریب شما استفاده شود. به عنوان مثال، اگر داده های شما از نوع داده های ترتیبی هستند، می توانید از روش تبدیل ArcSin نیز استفاده کنید. روش دیگری که می توانید از آن استفاده کنید، Reciprocal نامیده می شود. این روش اساساً به این صورت انجام می شود: 1/xکه x متغیر وابسته شماست.

Visually Inspect the Distribution of Your Variables

```
df["battery_power"].hist(grid=False, figsize=(10, 6), bins=100);
```



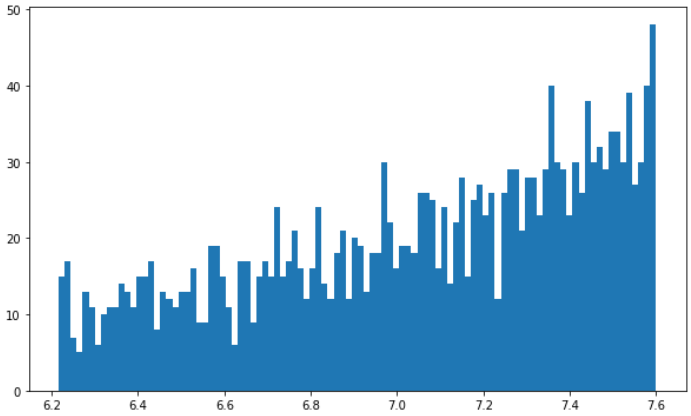

```
df["battery_power"].agg(['skew', 'kurtosis']).transpose()
```

```
skew        0.031898
kurtosis   -1.224144
Name: battery_power, dtype: float64
```

- Skewness
  - Fairly Symmetrical -0.5 to 0.5
  - Moderate Skewed -0.5 to -1.0 and 0.5 to 1.0
  - Highly Skewed < -1.0 and > 1.0

```
new_df = df
new_df.insert(len(new_df.columns), 'battery_power_log', np.log(df['battery_power']))
```

```
new_df["battery_power_log"].hist(grid=False, figsize=(10, 6), bins=100);
```




```
new_df.head()
```

fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range	battery_power_log
1	0	7	0.6	188	2	...	756	2549	9	7	19	0	0	1	1	6.735780
0	1	53	0.7	136	3	...	1988	2631	17	3	7	1	1	0	2	6.928538
2	1	41	0.9	145	5	...	1716	2603	11	2	9	1	1	0	2	6.333280
0	0	10	0.8	131	6	...	1786	2769	16	8	11	1	0	0	2	6.421622
13	1	44	0.6	141	2	...	1212	1411	8	2	15	1	1	0	1	7.507141

```
# train test split dataset
x_train, x_test, y_train, y_test = train_test_split(new_x, binary_y, test_size=0.2, random_state=79)
```

```
# Create model
model = LogisticRegression(solver='newton-cg', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9775
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       192
           1       0.97      0.99      0.98       208

    accuracy                           0.98       400
   macro avg       0.98      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

(د

New feature creation

- ph_vol = phone volume (cm^3)

```
new_df = df
new_df['ph_vol'] = df['m_dep']* df['sc_h']* df['sc_w']
```

```
new_df.head()
```

fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range	battery_power_log	ph_vol
1	0	7	0.6	188	2	...	2549	9	7	19	0	0	1	1	6.735780	37.8
0	1	53	0.7	136	3	...	2631	17	3	7	1	1	0	2	6.928538	35.7
2	1	41	0.9	145	5	...	2603	11	2	9	1	1	0	2	6.333280	19.8
0	0	10	0.8	131	6	...	2769	16	8	11	1	0	0	2	6.421622	102.4
13	1	44	0.6	141	2	...	1411	8	2	15	1	1	0	1	7.507141	9.6

```
# Seperate dataset into x(train set) , y(target set)
x = new_df.drop(["price_range"], axis=1)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
```

```
# Create model
model = LogisticRegression(solver='liblinear', max_iter=1000)
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.975
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       192
           1       0.97      0.98      0.98       208

    accuracy                           0.97       400
   macro avg       0.98      0.97      0.97       400
weighted avg       0.98      0.97      0.97       400
```

جواب سوال ۷:

الف)

```
def plot_confusionmatrix(y_train_pred,y_train,dom, classes):
    print(f'{dom} Confusion matrix')
    cf = confusion_matrix(y_train_pred,y_train)
    sns.heatmap(cf,annot=True,yticklabels=classes
                ,xticklabels=classes,cmap='Blues', fmt='g')
    plt.tight_layout()
    plt.show()
```

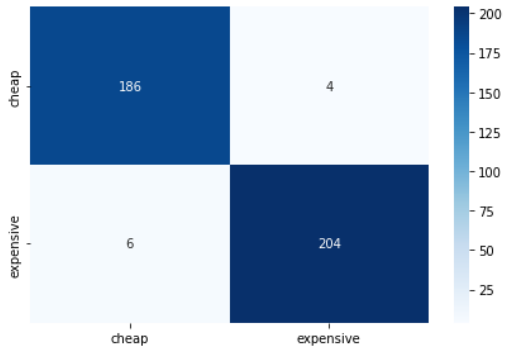
```
# Seperate dataset into x(train set) , y(target set)
x = df.drop(["price_range"], axis=1)
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
```

```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
dt_model.score(x_test, y_test)
```

```
0.945
```

```
classes = ['cheap','expensive']
plot_confusionmatrix(y_preds,y_test,'Test',classes)
```

```
Test Confusion matrix
```




```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       192
           1       0.97      0.98      0.98       208

    accuracy                           0.97       400
   macro avg       0.98      0.97      0.97       400
weighted avg       0.98      0.97      0.97       400
```

```
dt_model.get_params()
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

ب) تکنیک های هرس کردن

پیش هرس چیزی نیست جز توقف رشد درخت تصمیم در مراحل اولیه. برای آن می‌توانیم رشد درختان را با تعیین حد آستانه ها محدود کنیم. ما می توانیم پارامترهایی مانند max_depth، min_samples و غیره را محدود کنیم.

یک راه موثر برای انجام این کار این است که بتوانیم آن پارامترها را به صورت شبکه ای جستجو کنیم و مقادیر بهینه ای را انتخاب کنیم که عملکرد بهتری در داده های آزمایشی داشته باشد.

max_depth : حداکثر عمق درخت تصمیم

min_sample_split : حداقل تعداد نمونه های مورد نیاز برای تقسیم یک گره داخلی:

min_samples_leaf : حداقل تعداد نمونه مورد نیاز برای قرار گرفتن در یک گره برگ.

```
params = {'max_depth': [2,4,6,8,10,12],
          'min_samples_split': [2,3,4,5,6],
          'min_samples_leaf': [1,2,3,4,5,6,7,8,9]}

clf = DecisionTreeClassifier()
gcv = GridSearchCV(estimator=clf,param_grid=params)
gcv.fit(x_train,y_train)
model = gcv.best_estimator_
print(gcv.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=2, min_samples_split=5)
```

```
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

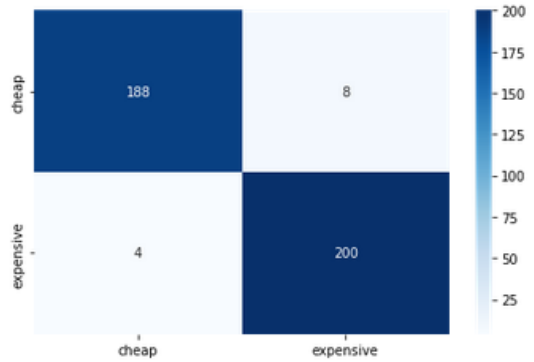
```
0.97
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       192
           1       0.98      0.96      0.97       208

    accuracy                           0.97       400
   macro avg       0.97      0.97      0.97       400
weighted avg       0.97      0.97      0.97       400
```

```
classes = ['cheap','expensive']
plot_confusionmatrix(y_preds,y_test,'Test',classes)
```

```
Test Confusion matrix
```



با توجه به گزارش طبقه بندی و امتیاز مدل و ماتریس درهم ریختگی میتوان به این نتیجه رسید که با هرس کردن درخت به نتیجه ای بهتر رسیده ایم.

جواب سوال ۸ :

```python
def backward_elimination(data, target,significance_level = 0.05):
    features = data.columns.tolist()
    while(len(features)>0):
        features_with_constant = sm.add_constant(data[features])
        p_values = sm.OLS(target, features_with_constant).fit().pvalues[1:]
        max_p_value = p_values.max()
        if(max_p_value >= significance_level):
            excluded_feature = p_values.idxmax()
            features.remove(excluded_feature)
        else:
            break
    return features
```

```python
backward_elimination(x, binary_y)
```

```
['battery_power', 'n_cores', 'px_height', 'px_width', 'ram']
```

```python
# Seperate dataset into x(train set) , y(target set)
x = df.drop(["price_range"], axis=1)
new_x = x[['battery_power', 'n_cores', 'px_height', 'px_width', 'ram']]
```

```python
x_train, x_test, y_train, y_test = train_test_split(new_x, binary_y, test_size=0.2, random_state=79)
```

```python
# Create model
model = LogisticRegression()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9925
```

```python
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

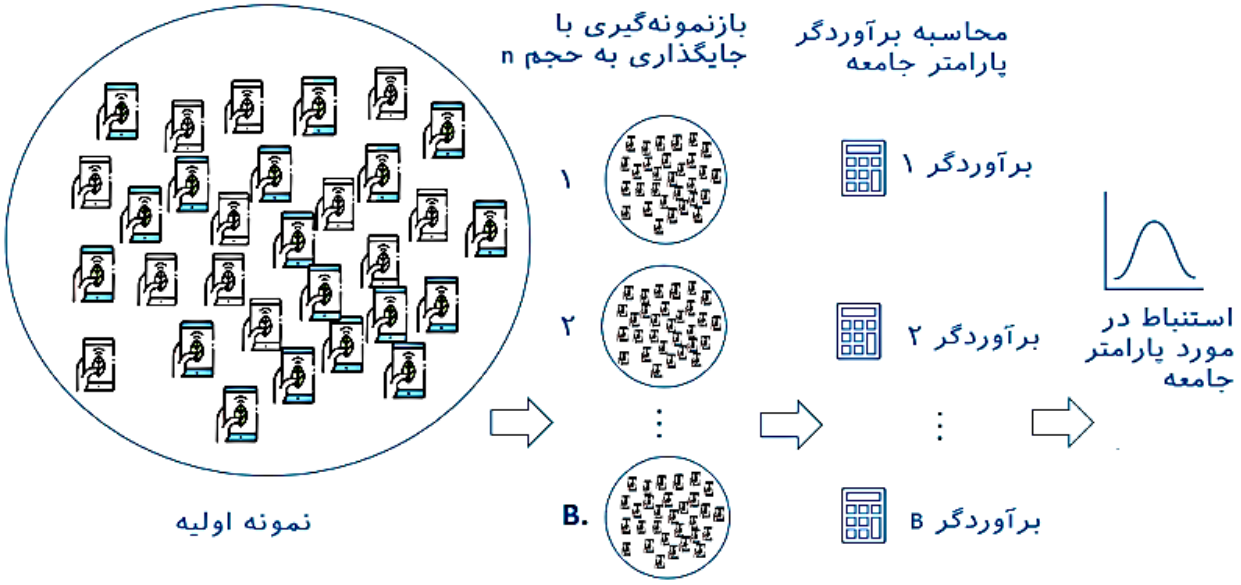
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       192
           1       1.00      0.99      0.99       208

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

جواب سوالات تشریحی:

جواب سوال ۱:

یکی از روش‌های قدرتمند رایانه‌ای در حوزه استنباط آماری، بوت استرپ (Bootstrap) است. این روش بدون در نظر گرفتن شرایط پیچیده، سعی در تخمین خطای برآوردگرها دارد. برای مثال برای ایجاد «فاصله‌های اطمینان» (confidence interval) «مدل‌های رگرسیونی» (Regression Models) و همچنین در حوزه «یادگیری ماشین» (Machine Learning) برای تخمین خطای مدل، می‌توان از روش بوت‌استرپ استفاده کرد. بوت استرپ در لغت به معنی تسمه‌ای است که بر روی چکمه قرار داشته و پوشیدن آن را ساده‌تر می‌کند. در مباحث آماری نیز به منظور برآورد راحت‌تر و ساده‌تر پارامترهای جامعه آماری، از تکنیک بوت‌استرپ استفاده می‌شود.

روش بوت استرپ به کمک تکنیک نمونه‌گیری با جایگذاری (بازنمونه‌گیری) سعی می‌کند که بهترین برآورد را برای خطای برآوردگرها با توجه به حجم نمونه محدود، بدست آورد. البته نمونه‌های حاصل از بازنمونه‌گیری، مستقل از یکدیگر هستند. مراحل مربوط به تکنیک بوت‌استرپ را می‌توان به صورت نمادین در زیر مشاهده کرد.



جواب سوال ۲:

در مورد 5x2 cross validation میتوان گفت که ارزیابی مدل با ۵ بار تکرار و k=2 یعنی تقسیم دیتاست به دو قسمت انجام می گردد. 5x2 cross validation توسط مقاله تست های آماری تقریبی برای مقایسه الگوریتم های یادگیری طبقه بندی نظارت شده توسط دیتریش به عنوان راهی برای به دست آوردن نه تنها یک تخمین خوب از خطای تعمیم، بلکه یک تخمین خوب از واریانس آن خطا رایج شد. (به منظور انجام تست های آماری)

جواب سوال ۳:

الگوریتم های مورد استفاده در درخت تصمیم

کتابخانه‌های مختلف زبان‌های برنامه‌نویسی مختلف از الگوریتم‌های پیش‌فرض خاصی برای ساختن درخت تصمیم استفاده می‌کنند، اما درک تفاوت بین الگوریتم‌های مورد استفاده برای یک دانشمند داده کاملاً نامشخص است. در اینجا به آن الگوریتم ها خواهیم پرداخت.

ID3

ID3 با در نظر گرفتن کل مجموعه S به عنوان گره ریشه یک درخت تولید می کند. سپس روی هر ویژگی تکرار می‌شود و داده‌ها را به قطعاتی تقسیم می‌کند که به عنوان زیرمجموعه شناخته می‌شوند تا آنتروپی یا به دست آوردن اطلاعات آن ویژگی را محاسبه کند. پس از تقسیم، الگوریتم به هر زیرمجموعه با در نظر گرفتن آن دسته از ویژگی‌هایی که قبلاً در موارد تکراری گرفته نشده‌اند،

متوسل می‌شود. این یک الگوریتم ایده آل نیست زیرا به طور کلی داده ها را بیش از حد برازش می دهد و در متغیرهای پیوسته، تقسیم داده ها می تواند زمان بر باشد.

C4.5

در مقایسه با ID3 بسیار پیشرفته است زیرا داده هایی را که نمونه های طبقه بندی شده در نظر می گیرد. تقسیم بر اساس به دست آوردن اطلاعات نرمال شده انجام می شود و ویژگی دارای بالاترین بهره اطلاعاتی تصمیم می گیرد. برخلافID3 ، می‌تواند ویژگی‌های پیوسته و گسسته را بسیار کارآمد و پس از ساختن درخت مدیریت کند. با حذف تمام شاخه های کم اهمیت هرس می شود.

CART

CART می تواند هم وظایف طبقه بندی و هم رگرسیون را انجام دهد و با در نظر گرفتن شاخص جینی بر خلاف ID3 یا C4.5 که از نسبت به دست آوردن اطلاعات و بهره برای تقسیم استفاده می کند، نقاط تصمیم را ایجاد می کند. برای تقسیم، CART از یک الگوریتم حریصانه پیروی می کند که هدف آن فقط کاهش تابع هزینه است. برای طبقه بندی از تابع هزینه مانند شاخص جینی برای نشان دادن خلوص گره های برگ استفاده می شود. برای رگرسیون، مجموع مجذور خطا توسط الگوریتم به عنوان تابع هزینه انتخاب می شود تا بهترین پیش بینی را پیدا کند.

CHAID

CHAIDیا-Chi آشکارساز تعامل خودکار مربعی فرآیندی است که می تواند با هر نوع متغیری اعم از اسمی، ترتیبی یا پیوسته سروکار داشته باشد. در درخت رگرسیون از آزمون F و در درختان طبقه بندی از آزمون Chi-Square استفاده می کند. در این تحلیل، پیش‌بینی‌کننده‌های پیوسته به تعداد مساوی مشاهدات تقسیم می‌شوند تا زمانی که یک نتیجه به دست آید. در مقایسه با سایر الگوریتم ها در مسائل دنیای واقعی بسیار کمتر مورد استفاده قرار می گیرد.

MARS

MARS یا خطوط رگرسیون تطبیقی چند متغیره، تحلیلی است که به‌ویژه در مسائل رگرسیون زمانی که داده‌ها عمدتاً ماهیت غیرخطی دارند، اجرا می‌شود.

جواب سوال ۴:

```
from sklearn.tree import DecisionTreeClassifier
```

```
# Seperate dataset into x(train set) , y(target set)
x = df.drop(["price_range"], axis=1)
x_train, x_test, y_train, y_test = train_test_split(x, binary_y, test_size=0.2, random_state=79)
```

```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
dt_model.score(x_test, y_test)
```

```
0.945
```

```
y_preds = model.predict(x_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       192
           1       0.97      0.98      0.98       208

    accuracy                           0.97       400
   macro avg       0.98      0.97      0.97       400
weighted avg       0.98      0.97      0.97       400
```

جواب سوال ۵:

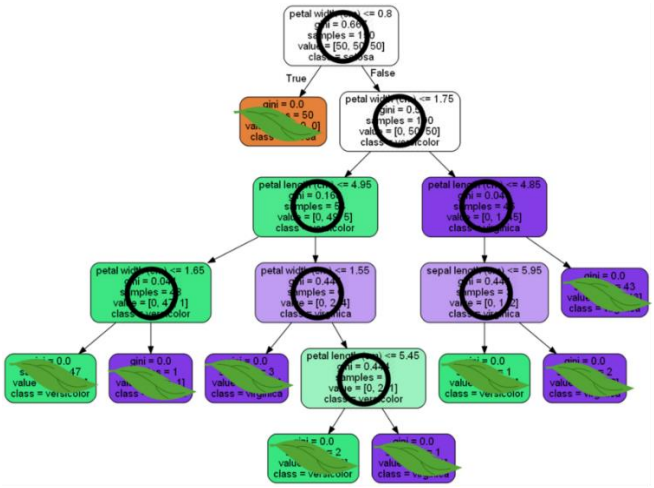
```
dt_model.get_params()
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

Criterion : این پارامتر نحوه اندازه گیری ناخالصی یک شکاف را تعیین می کند. مقدار پیش‌فرض «gini» است، اما می‌توانید از «آنتروپی» به عنوان معیار ناخالصی نیز استفاده کنید.

Splitter : به این ترتیب درخت تصمیم ویژگی ها را برای یک تقسیم جستجو می کند. مقدار پیش فرض روی "بهترین" تنظیم شده است. یعنی برای هر گره الگوریتم تمام ویژگی ها را در نظر می گیرد و بهترین تقسیم را انتخاب می کند. اگر تصمیم دارید که پارامتر تقسیم کننده را روی "تصادفی" تنظیم کنید، یک زیرمجموعه تصادفی از ویژگی ها در نظر گرفته می شود. سپس تقسیم توسط بهترین ویژگی در زیر مجموعه تصادفی انجام می شود. اندازه زیر مجموعه تصادفی با پارامتر max_features تعیین می شود. این جایی است که یک جنگل تصادفی شکل می گیرد.

max_depth : حداکثر عمق درخت را تعیین می کند. در مورد ما، ما از عمق دو برای ایجاد درخت تصمیم خود استفاده می کنیم. مقدار پیش فرض روی هیچ تنظیم شده است. این اغلب منجر به درختان تصمیم گیری بیش از حد برازش می شود. پارامتر عمق یکی از راه هایی است که می توانیم درخت را منظم کنیم، یا راه رشد آن را برای جلوگیری از برازش بیش از حد محدود کنیم. در شکل پایین می توانید ببینید اگر عمق درخت را تنظیم نکنید چه اتفاق می افتد !



درخت تصمیم با رشد کامل : در درخت نشان داده شده در بالا، هیچ یک از پارامترها تنظیم نشده است. درخت به طور کامل تا عمق پنج رشد می کند. هشت گره و نه برگ وجود دارد. محدود نکردن رشد درخت تصمیم ممکن است به برازش بیش از حد منجر شود.

min_samples_split : حداقل تعداد نمونه هایی که یک گره باید داشته باشد تا بتوان تقسیم بندی را در نظر گرفت. مقدار پیش فرض دو است. می توانید از این پارامتر برای منظم کردن درخت خود استفاده کنید.

min_samples_leaf : حداقل تعداد نمونه مورد نیاز برای در نظر گرفتن گره برگ. مقدار پیش فرض روی یک تنظیم شده است. از این پارامتر برای محدود کردن رشد درخت استفاده کنید.

max_features: تعداد ویژگی هایی که هنگام جستجوی بهترین تقسیم باید در نظر بگیرید. اگر این مقدار تنظیم نشود، درخت تصمیم تمام ویژگی های موجود را برای ایجاد بهترین تقسیم در نظر می گیرد. بسته به برنامه شما، اغلب ایده خوبی است که این پارامتر را تنظیم کنید.

جواب سوال ۶:

درخت‌های تصمیم، ساختارهای داده درختی هستند که با استفاده از الگوریتم‌های یادگیری به منظور طبقه‌بندی و رگرسیون تولید می‌شوند. یکی از رایج ترین مشکلات هنگام یادگیری درخت تصمیم، یادگیری اندازه بهینه درخت به دست آمده است که منجر به دقت بهتر مدل می شود.

درختی که دارای شاخه‌ها و لایه‌های بیش از حد است می‌تواند منجر به تطبیق بیش از حد داده‌های آموزشی شود. هرس درخت تصمیم به جلوگیری از برازش بیش از حد داده های آموزشی کمک می کند تا مدل ما به خوبی به داده های دیده نشده تعمیم یابد. هرس درخت تصمیم به معنای حذف درخت فرعی است که زائد است و شکاف مفیدی ندارد و آن را با یک گره برگ جایگزین می کنیم.

هرس درخت تصمیم را می توان به دو نوع تقسیم کرد: قبل از هرس و پس از هرس.

پیش هرس، همچنین به عنوان قانون توقف زودهنگام شناخته می شود، روشی است که در آن ساخت زیردرخت در یک گره خاص پس از ارزیابی برخی اقدامات متوقف می شود. این اقدامات می تواند ناخالصی جینی یا به دست آوردن اطلاعات باشد. در پیش هرس، شرایط هرس را بر اساس اقدامات فوق در هر گره ارزیابی می کنیم.

پیش هرس مزیت این را دارد که سریعتر و کارآمدتر باشد زیرا از ایجاد زیردرختهای بیش از حد پیچیده که بیش از حد بر داده های آموزشی منطبق هستند جلوگیری می کند. با این حال، در پیش هرس، رشد درخت پیش از موعد با شرایط توقف ما متوقف می شود.

پس هرس، همانطور که از نام آن پیداست، پس از هرس به معنای هرس کردن پس از ساخت درخت است. شما درخت را به طور کامل با استفاده از الگوریتم درخت تصمیم خود رشد می دهید و سپس درختان فرعی درخت را به صورت پایین به بالا هرس می کنید. شما از گره تصمیم گیری پایین شروع می کنید و بر اساس معیارهایی مانند ناخالصی جینی یا افزایش اطلاعات، شما تصمیم می گیرید که آیا این گره تصمیم را حفظ کنید یا آن را با یک گره برگ جایگزین کنید. برای مثال، فرض کنید می‌خواهیم درخت‌های فرعی را هرس کنیم که کمترین اطلاعات را به دست می‌آورند. هنگام تصمیم گیری برای گره برگ، می خواهیم بدانیم اگر الگوریتم درخت تصمیم ما این گره تصمیم را ایجاد نمی کرد، چه برگه ای ایجاد می کرد.

الگوریتم های هرس

الگوریتم های هرس زیادی وجود دارد. در زیر سه نمونه از الگوریتم های هرس آورده شده است.

هرس با کسب اطلاعات

ما می توانیم درخت تصمیم خود را با استفاده از اطلاعات بدست آمده در هر دو مرحله پس از هرس و قبل از هرس هرس کنیم. در پیش هرس، بررسی می کنیم که آیا افزایش اطلاعات در یک گره خاص از حداقل بهره بیشتر است یا خیر. در پس ازهرس، درختان فرعی را با کمترین اطلاعات هرس می کنیم تا به تعداد برگ دلخواه برسیم.

هرس خطای کاهش یافته (REP)

REP متعلق به دسته Post-Pruning است. در REP، هرس با کمک یک مجموعه اعتبارسنجی انجام می شود. در REP، همه گره ها برای هرس به صورت پایین به بالا ارزیابی می شوند. اگر درخت هرس شده بدتر از درخت اصلی در مجموعه اعتبار سنجی

نباشد، یک گره هرس می شود. درخت فرعی در گره با یک گره برگ جایگزین می شود که رایج ترین کلاس را به خود اختصاص می دهد.

هرس با پیچیدگی هزینه

هرس با پیچیدگی هزینه نیز در دسته پس از هرس قرار می گیرد. هرس هزینه- پیچیدگی با محاسبه امتیاز درخت بر اساس مجموع مربعات باقیمانده (RSS) برای زیردرخت، و جریمه پیچیدگی درخت که تابعی از تعداد برگ‌های درخت فرعی است، کار می‌کند. جریمه پیچیدگی درخت، تفاوت در تعداد برگ ها را جبران می کند.

$$TreeScore=RSS+aT$$

(آلفا) یک فراپارامتر است که با استفاده از اعتبارسنجی متقاطع پیدا می‌کنیم و T تعداد برگ‌های درخت فرعی است.

ما امتیاز درخت را برای همه زیردرخت های درخت تصمیم محاسبه می کنیم و سپس زیردرختی را با کمترین امتیاز درخت انتخاب می کنیم. با این حال، می توانیم از معادله مشاهده کنیم که مقدار آلفا انتخاب زیردرخت را تعیین می کند. مقدار آلفا با استفاده از اعتبارسنجی متقاطع یافت می شود. ما فرآیند بالا را برای مقادیر مختلف آلفا تکرار می کنیم،

که دنباله ای از درختان را به ما می دهد. مقدار آلفا که به طور متوسط کمترین امتیاز درختی را می دهد، مقدار نهایی آلفا است. در نهایت، درخت تصمیم هرس شده ما درختی خواهد بود که با مقدار نهایی آلفا مطابقت دارد.

جواب سوال ۷:

در بحث بایاس و واریانس به توضیح مختصر می توان نوشت که اگر ما تعداد داده ای داشته باشیم که مدل آموزش دیده ما برای آن ها تخمین انجام گیرد و ما مقدار واقعی این مقادیر تخمینی را داشته باشیم امید ریاضی ما همان میانگین مقادیر واقعی و تعریف بایاس اختلاف میانگین مقادیر تخمینی و امید ریاضی ما می باشد و به نوعی بایاس هر چقدر کمتر باشد در کل داده های ما به نوعی به داده های واقعی نزدیک تر هستند ولی این مورد به تنهایی کافی نیست چون بایاس فقط اختلاف میانگین ها می باشدو نمود کل جامعه آماری ما نیست پس معیار دیگری نیز باید باشد که اختلاف تک تک مقادیر واقعی و تخمینی از همدیگر را کنترل کند و این معیار، واریانس است . واریانس به نوعی نزدیک بودن تک تک مقادیر تخمینی به مقادیر اصلی را نشان میدهد پس با ترکیب بایاس که نزدیک بودن کل مقادیر ها از نظر مقداری به هم و واریانس که نزدیک بودن پراکندگی مقادیر نسبت هم را نشان میدهد میتواند معیاری باشد که پیچیدگی مدل ما را بهینه کند و این در حالتی است که مقدار بایاس کم و مقدار واریانس هم کم باشد.

جواب سوال ۸:

آزمون مناسب برای ارزیابی اهمیت آماری بسته به آنچه که مدل یادگیری ماشین شما پیش‌بینی می‌کند، توزیع داده‌های شما، و اینکه آیا پیش‌بینی‌ها را در مورد موضوعات مورد نظر مقایسه می‌کنید یا نه، متفاوت است.

آزمون آماری فرضیه صفر

یک آزمون آماری فرضیه صفر برای مقایسه دو نمونه از داده ها و محاسبه اطمینان آماری استفاده می شود که تفاوت درک شده در یک ویژگی ریاضی (به عنوان مثال مقادیر میانگین های مختلف) در جامعه وسیع تر مشاهده می شود. اگرچه انتقادات مداومی از آزمون اهمیت فرضیه صفر وجود دارد، هنوز هم متداول ترین تکنیک مورد استفاده برای این اعتبار است.

فرضیه صفر

فرضیه صفر عبارت است از اینکه تفاوتی بین توزیع دو نمونه داده وجود ندارد. که هر گونه واریانسی به دلیل نویز یا شانس دیده می شود. از نظر یادگیری ماشین، فرضیه صفر ما این است که معیارهای عملکرد برابر هستند، هر سود یا ضرر کوچک، مشاهده شده از نظر آماری معنی دار نیست.

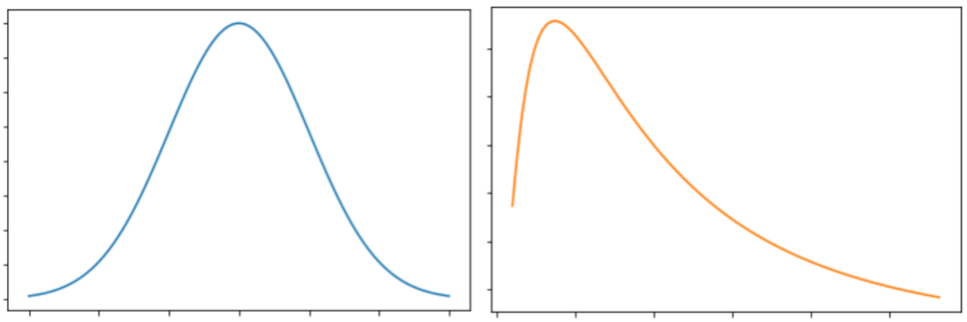
P-Value

این آزمون نشان می‌دهد که آیا باید فرضیه صفر مبنی بر عدم وجود تفاوت را رد کنیم یا نتوانیم رد کنیم. آزمون های آماری مورد بحث در زیر تمایل به ارائه یک آمار آزمون و یک مقدار p دارند. اگر مقدار p زیر یک آستانه معین باشد (اغلب0.05) می‌توانیم فرضیه صفر را رد کنیم و استنباط کنیم که این تفاوت از نظر آماری معنی‌دار است.

مفروضات

توزیع داده ها

هنگام استفاده از آزمون‌های فرضیه برای مدل‌های رگرسیون، مهم است که تشخیص دهید آیا باید از آزمون پارامتریک یا ناپارامتریک استفاده کنید. آزمون‌های پارامتری ترجیح داده می‌شوند زیرا معمولاً قدرت آماری بیشتری نسبت به آزمون ناپارامتریک دارند. این بدان معنی است که آنها در صورت وجود، احتمال بیشتری برای تشخیص یک اثر آماری معنی دار دارند. با این حال، این آزمون ها فرض می کنند که داده ها به طور معمول توزیع شده اند. اگر این فرض درست نباشد، باید از آزمون ناپارامتریک استفاده شود.



وقتی مقادیر خطا را مقایسه می کنید، به احتمال زیاد به سمت چپ منحرف شده اند: اکثر خطاها کوچک هستند و با افزایش مقدار خطا، فرکانس کاهش می یابد. اگر قرار بود داده های خود را رسم کنید، ممکن است بتوانید این را به صورت بصری تفسیر کنید، اما تست های نرمال بودن نیز وجود دارد که می توانید از آنها برای تأیید ریاضی استفاده کنید.

مشاهدات جفت یا جفت نشده

اگر داده‌ها از افراد مشابه برای هر مدل جمع‌آوری شده باشد، به آن "جفت" گفته می‌شود. در یادگیری ماشینی، این بدان معناست که داده‌های آزمایش برای خط پایه و مدل آموزش‌دیده یکسان است. داده های جمع آوری شده از دو گروه مستقل به عنوان "جفت نشده" نامیده می شود. این در تحقیقات علمی رایج است، یا تست A/B، زمانی که یک گروه کنترل را با یک گروه درمانی مقایسه می کنید.

آزمون های آماری مناسب

Regression

اگر مدل یادگیری ماشین شما مقادیر عددی را پیش‌بینی می‌کند، معیار خطا یکی از موارد زیر است:

میانگین مربع خطا (MSE)

میانگین خطای مطلق (MAE)

میانگین درصد مطلق خطا (MAPE)

میانگین وزنی درصد خطای مطلق (WMAPE)

هنگامی که از آزمون فرضیه برای مقایسه میانگین خطاها استفاده می کنید، به احتمال اینکه خطاها از توزیع یکسانی می آیند نگاه می کنید. اگر احتمال به اندازه کافی کم باشد، تفاوت بین دو مقدار میانگین خطا از نظر آماری معنادار است.

	Unpaired	Paired
Parametric	Independent t-test (Student's or Welch's)	Paired t-test (Student or Welch's)
Non-parametric	Mann-Whitney U test	Wilcoxon signed-rank test

برای داده هایی که به طور معمول توزیع شده اند، دو آزمون مناسب عبارتند از آزمون t استودنت یا آزمون t ولش. هر دو آزمایش می‌کنند که دو جامعه میانگین یکسانی دارند، اما آزمون t ولش از نظر وجود تفاوت در واریانس یا اندازه نمونه قابل اعتمادتر است. بسته به جفت یا جفت نبودن مشاهدات دو پیاده سازی وجود دارد.

دو آزمون ناپارامتریک یا آزمون من ویتنی U یا آزمون رتبه علامت دار ویلکاکسون هستند، بسته به اینکه مشاهدات زوجی باشند یا نه.

طبقه بندی

اگر مدل یادگیری ماشین شما پیش‌بینی می‌کند که یک نمونه متعلق به کدام کلاس است، معیارهای خطا یکی از موارد زیر است:

Accuracy

Precision

Recall

F1-score

اگرچه موضوع کار شما باید تصمیم بگیرد که کدام معیار باید اولویت بندی شود، رایج ترین تکنیک های مورد استفاده در مدل های طبقه بندی به سادگی ارزیابی می کنند که آیا نسبت خطاها به طور قابل توجهی متفاوت است یا خیر. اگر صراحتاً مایل به آزمایش اهمیت آماری Precision یا F1-score هستید، تحقیقات بیشتری لازم است.

Binary	Multi-class
McNemar’s test	Stuart-Maxwell test

مدل‌های طبقه‌بندی می‌توانند باینری باشند (به‌عنوان مثال، تکان‌خورده یا ناسازگار)، یا چند طبقه (مانند ورزش، سیاست یا علم). برای مقایسه مدل طبقه بندی باینری، آزمون مناسب آزمون مک نمار است، در حالی که برای چند کلاس آزمون استوارت-مکسول است.

جواب سوال ۹:

ضریب همبستگی متیو چیست؟

ضریب همبستگی متیو که به اختصار MCC نیز نامیده می شود توسط برایان متیوز در سال 1975 اختراع شد. MCC یک ابزار آماری است که برای ارزیابی مدل استفاده می شود. وظیفه آن اندازه گیری یا اندازه گیری تفاوت بین مقادیر پیش بینی شده و مقادیر واقعی است و معادل Chi-square آمار برای یک جدول احتمالی ۲ x ۲ .

فرمول ضریب همبستگی متیو

MCC بهترین معیار طبقه بندی تک مقداری است که به خلاصه کردن ماتریس سردرگمی یا ماتریس خطا کمک می کند. یک ماتریس سردرگمی دارای چهار موجودیت است:

- نکات مثبت واقعی (TP)
- منفی های واقعی (TN)
- موارد مثبت کاذب (FP)
- منفی کاذب (FN)

و با فرمول محاسبه می شود:

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

اگر پیش‌بینی نرخ‌های خوبی را برای هر چهار مورد از این موجودیت ها به دست آورد، گفته می‌شود که معیار قابل اعتمادی است که نمرات بالایی ایجاد می‌کند. و برای مطابقت با اکثر ضرایب همبستگی، MCC نیز بین 1+ و 1- به صورت زیر است:

۱+ بهترین توافق بین مقادیر پیش بینی شده و واقعی است.

۰ توافقی نیست به این معنی که پیش بینی با توجه به واقعیات تصادفی است

مزایای MCC نسبت به امتیاز F1

زمینه:

برای ارزیابی طبقه بندی های دوتایی و ماتریس های سردرگمی آنها، محققان علمی می توانند بر اساس هدف آزمایشی که در حال بررسی هستند، از چندین نرخ آماری استفاده کنند. علیرغم اینکه یک مسئله حیاتی در یادگیری ماشین است،

هنوز هیچ اجماع گسترده ای در مورد یک اقدام منتخب انتخابی یکپارچه حاصل نشده است. دقت و امتیاز F1 محاسبه‌شده بر روی ماتریس‌های سردرگمی یکی از محبوب‌ترین معیارهای پذیرفته‌شده در وظایف طبقه‌بندی باینری بوده است (و هنوز هم هستند). با این حال، این معیارهای آماری می توانند به طور خطرناکی نتایج متورم بیش از حد خوش بینانه را نشان دهند.به ویژه در مورد مجموعه داده های نامتعادل.

نتایج: در عوض، ضریب همبستگی متیوز (MCC)، نرخ آماری قابل اعتمادتری است که تنها در صورتی امتیاز بالایی ایجاد می‌کند که پیش‌بینی نتایج خوبی در هر چهار دسته ماتریس سردرگمی (مثبت واقعی، منفی کاذب، منفی درست و نادرست) به دست آورد. موارد مثبت)

متناسب با اندازه عناصر مثبت و اندازه عناصر منفی در مجموعه داده است.

نتیجه‌گیری: در این تمرین نشان می‌دهیم که چگونه MCC در ارزیابی طبقه‌بندی‌های باینری امتیاز آموزنده‌تر و درست‌تری نسبت به دقت و امتیاز F1 ایجاد می‌کند، ابتدا با توضیح ویژگی‌های ریاضی، و سپس دارایی MCC در شش مورد استفاده مصنوعی و در یک سناریوی ژنومیک واقعی. ما معتقدیم که ضریب همبستگی متیوز باید به دقت و امتیاز F1 در ارزیابی وظایف طبقه‌بندی باینری توسط همه جوامع علمی ترجیح داده شود.

تمامی مراحل با ذکر کامنت هم به صورت انگلیسی هم فارسی به طور کامل در فایل ژوپیتر برای این تمرین انجام شده است.

باتشکر محمدسعید حیدری