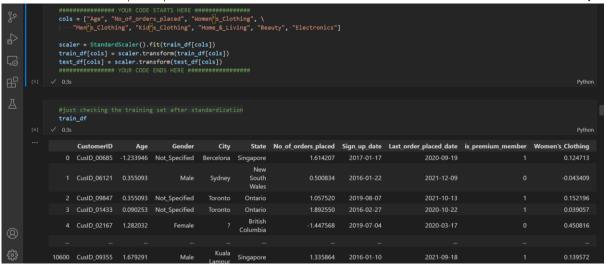
۱٫۱) برای از بین بردن مقادیر NaN از forward fill استفاده شده است که هر جا که به NaN میرسد، مقدار موجود در ردیف قبلی را جایش میگذارد. در ادامه هم چک شد که هیچ NaNای وجود نداشته باشد (اگر ردیف اول NaN بود این مشکل وجود میداشت که در اون صورت میشد یک بار نیز backward fill انجام داد)

۱٫۲) ستونهایی که عددی هستند را انتخاب کرده (چون StandardScaler روی string کار نمیکند و همچنین اب وجود اینکه میشد برای هر رشته یک مقدار عددی جنریت کرد، انجام این کار روی این داده ها معنی نداشت به جز مثلا تاریخها) و جون باید train و test هردو نسبت به train نرمالایز شوند، fit را روی train انجام میدهیم.



۱٫۳) در این قسمت هم روی ستونها خواسته شده one hot encoding را پیاده سازی میکنیم.

昭		cols_to_be_encoded = ['Gender', 'State']											
Д		######################################											
	######################################												Python
		train_df											
													Python
				CustomerID	Age	No_of_orders_placed	Sign_up_date	Last_order_placed_date	is_premium_member	Women's_Clothing	Men's_Clothing	Kid's_Clothing	Home_
				CusID_09265	1.944131	1.614207	2019-04-17	2021-05-25		0.117018	0.269042	1.073437	
8				CusID_05791	0.355093	0.500834	2017-06-23	2021-07-30		-0.039028	1.288364	-1.767676	
				CusID_07705	2.473810	1.335864	2017-09-29	2021-01-06		0.388687	-0.977850	1.562905	
565				CusID_08159	-0.439427	1.057520	2018-11-06	2020-09-02		0.313424	-1.484451	0.380756	

۲٫۱) برای این قسمت، چون الگوریتمهای ML همانطور که گفته شد با رشته نمیتوانند کار کنند، ستوانهای رشته اول drop) شدند و سپس split انجام شده است.

برای پیاده سازی، الگوریتم KNN انتخاب شده است که در حقیقت برای اینکه هر نقطه را در کلاس خود قرار دهد (ابنجا کلاسهایمان دوست داشتن IU قدیم یا جدید است) فاصله آن تا نقاط دیگر را محاسبه میکند (فاصله اقلیدسی یا منهتن یا.. که اینجا اقلیدسی انتخاب شده است) و با توجه به اینکه بین k نقطه نزدیکتر، بیشترشان چه کلاسی دارند، کلاس این نقطه را پیش بینی میکند. برای محاسبه فاصله نقاط نیز فاصله اقلیدسی هر مقدار در هر ستون را با مقدار آن ستون در نقاط موجود در training set

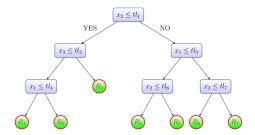
```
############## YOUR CODE STARTS HERE ###############
#implementing KNN from scratch
#note that Scikit-learn uses a KD Tree or Ball Tree to compute nearest neighbors in O[N log(N)] time
#while this algorithm is a direct approach including nested lopps that requires O[N^2] time
#therefore processing 900 points in x_val takes about 9 minutes
def eucledian_dist(p1, p2):
   p = p2 - p1
   return np.sum(p*p) #not using np.sqr because it doesn't change the result + it slows down the process
def KNN_predict(x_train, y_train, x_val, k):
   y_vals = []
   for i in range(len(x_val)):
       point_dist = []
        for j in range(len(x train)): #number of rows in x train
            distances = eucledian_dist(np.array(x_val.iloc[i]), np.array(x_train.iloc[j]))
            point_dist.append(distances)
       point_dist = np.array(point_dist)
       dist = np.argsort(point_dist)[:k]
       vals = y_train[dist]
       #mode of val
       v = mode(vals)
       v = v.mode[0]
       y_vals.append(v)
   return y_vals
y_pred = KNN_predict(X_train, y_train, X_val, 5)
np.array(y_pred)
```

```
### - YOUR - CODE - ENDS - HERE - ###################
Output exceeds the size limit. Open the full output data in a text editor
        'New_UI', 'New_UI', 'New_UI', 'Old_UI', 'Old_UI', 'Old_UI',
                  'Old UI'.
                             'New UI'.
                                        'New UI'.
                                                  'Old UI'.
                                        'Old UI'.
                  '01d_UI'
                             'New_UI',
                                        'Old_UI',
                                        'Old UI'.
                              'New_UI', 'Old_UI',
                                                   'New_UI',
        'Old_UI', 'New_UI', 'Old_UI', 'New_UI',
        'Old_UI', 'New_UI', 'New_UI', 'New_UI', 'New_UI', 'New_UI'
```

۲٫۲) در این قسمت الگوریتم اول همان KNN است که تفاوت جواب با الگوریتم پیاده سازی خودمان مشاهده شود. در بالای کد قسمت قبل نیز توضیح داده شده است چرا سرعت KNN در sklearn اصلا قابل مقایسه با کد ما نیست. تغییر k نیز میتواند روی دقت الگوریتم تاثیر بگذارد (نه باید خیلی بزرگ باشد نه خیلی کوچک)

```
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred2 = classifier.predict(X_val)
array(['New_UI', 'New_UI', 'Old_UI', 'Old_UI', 'Old_UI', 'New_UI',
         'Old_UI', 'Old_UI', 'New_UI', 'Old_UI', 'New_UI', 'Old_UI',
         'New_UI', 'Old_UI', 'New_UI', 'New_UI', 'Old_UI', 'Old_UI'
                                            'Old UI'.
                                 'New_UI',
                                            'New_UI',
                                                         'New_UI',
                                 'New UI'.
                                             'Old UI'.
                                                         'Old UI'
                    'New_UI',
                                 'New_UI', 'New_UI',
                                                         'New_UI',
         'Old_UI', 'New_UI', 'Old_UI', 'New_UI',
                                                         'New_UI',
```

برای الگوریتم دوم Decision Tree انتخاب شده است. در این الگوریتم برای هر مقدار از یک ستون، در درخت یک شاخه به وجود می آید که درخت را روی آن پایین برود و هروقت به برگ برسد، به مقدار موردنیاز برای پیش بینی رسیده است.



برای شکاف درخت از gini impurity استفاده میشود: در صورتی که به یک آیتم به طور تصادفی بر اساس توزیع دادهها کلاسی اختصاص داده شود، احتمال دستهبندی اشتباه را اندازه گیری می کند. به این شکل که اول داده ها را در امتداد هر عنصر منحصر به فرد تقسیم می کنیم، gini impurity را برای هر تقسیم مقدار هدف محاسبه می کنیم و سپس هر ناخالصی را بر اساس نمرات کلی وزن می کنیم. شکافی که کمترین ناخالصی وزنی را ایجاد می کند، شکافی است که برای تقسیم استفاده میشود.

DecisionTreeClassifier ورودیهایی دارد که روی دقت آن تاثیر میگذارند مثلا: min_samples_split نشان دهنده حداقل تعداد نمونه مورد نیاز برای تقسیم یک گره است. max_depth حداکثر عمق درخت است، اگر None باشد، گرهها تا min_samples_split داشته باشند، گسترش می یابند. و چند ورودی دیگر که زمانی که تمام برگها خالص شوند یا کمتر از max_depth داشته باشند، گسترش می یابند. و چند ورودی دیگر که در این کد فقط با اعطای مقادیر بین ۱ تا ۹ به accuracy، مقداری که بیشترین دقت را داشته است انتخاب شده است (دقت در قسمت بعدی محاسبه میشود و همان accuracy)

۲٫۳) در این قسمت اینکه هر تابع چه چیزی را محاسبه میکند در کد کامنت شده است.

```
### Securacy/months in the Classifier correct?

print(*Accuracy): ", accuracy_score(y_sal, y_pred))

print(*Teccional: ", accuracy_score(y_sal, y_pred))

print(*Teccional: ", accuracy_score(y_sal, y_pred))

print(*Teccional: ", accuracy_score(y_sal, y_pred))

print(*Teccional: ", accuracy_score(y_sal, y_pred), acc
```