به نام خدا امیرحسین باوند ۹۳۳۱۰۲۸

قسمت یک :

در این قسمت ابتدا رابطه داده ها را با هم بررسی می کنیم .

ابتدا کوریلیشن تمام داده ها را با هم بررسی می کنیم .

که به داده های زیر میرسیم .

```
C:\Users\pc\Anaconda3\python.exe C:/Term6/DM/HW5_FRAUD/test.py
```

\	amount	hour_a	zip	customerAttr_a	field_a
amount	1.000000	0.016795	-0.007203	0.058196	-0.096192
hour_a	0.016795	1.000000	0.037187	0.012057	-0.001555
zip	-0.007203	0.037187	1.000000	-0.024513	0.065908
customerAttr_a	0.058196	0.012057	-0.024513	1.000000	-0.001656
field_a	-0.096192	-0.001555	0.065908	-0.001656	1.000000
field_b	0.005488	0.018180	-0.009157	0.002764	0.055410
hour_b	0.017228	0.993949	0.038781	0.012216	-0.002072
flag_a	-0.213758	-0.042825	0.028345	-0.129299	-0.017339
total	1.000000	0.016795	-0.007203	0.058196	-0.096192
field_c	-0.084633	0.014707	-0.019354	0.002879	0.113244
field_d	0.062259	0.016893	-0.007174	0.062927	0.020189
indicator_a	-0.029256	0.005947	0.073175	0.000306	0.023651
indicator_b	-0.019238	-0.007011	0.014159	-0.008071	0.010559
flag_b	0.110985	-0.017504	-0.018511	-0.004262	0.028043
flag_c	0.016743	-0.013285	-0.012477	-0.017805	0.057747
flag_d	-0.020267	-0.002429	0.004486	-0.027808	-0.008745

flag_e	-0.140758	-0.027809	0.018130	0.011753	0.123510	

\	field_b	hour_b	flag_a	total	field_c	field_d
amount	0.005488	0.017228	-0.213758	1.000000	-0.084633	0.062259
hour_a	0.018180	0.993949	-0.042825	0.016795	0.014707	0.016893
zip	-0.009157	0.038781	0.028345	-0.007203	-0.019354	-0.007174
customerAttr_a	0.002764	0.012216	-0.129299	0.058196	0.002879	0.062927
field_a	0.055410	-0.002072	-0.017339	-0.096192	0.113244	0.020189
field_b	1.000000	0.017555	-0.074339	0.005488	0.040068	0.059453
hour_b	0.017555	1.000000	-0.044476	0.017228	0.013854	0.017359
flag_a	-0.074339	-0.044476	1.000000	-0.213758	0.045064	-0.444587
total	0.005488	0.017228	-0.213758	1.000000	-0.084633	0.062259
field_c	0.040068	0.013854	0.045064	-0.084633	1.000000	0.021890
field_d	0.059453	0.017359	-0.444587	0.062259	0.021890	1.000000
indicator_a	0.004318	0.006489	0.007717	-0.029256	0.138208	0.002588
indicator_b	-0.009476	-0.007365	0.027672	-0.019238	-0.008878	-0.021262
flag_b	0.020034	-0.016498	-0.010917	0.110985	-0.058493	0.045699
flag_c	-0.005449	-0.013151	-0.004212	0.016743	-0.022408	0.045575
flag_d	-0.012880	-0.002922	0.105650	-0.020267	-0.040561	-0.084106
flag_e	0.018027	-0.027886	-0.037609	-0.140758	-0.008101	-0.037919

\	indicator_a	indicator_b	flag_b	flag_c	flag_d
amount	-0.029256	-0.019238	0.110985	0.016743	-0.020267
hour_a	0.005947	-0.007011	-0.017504	-0.013285	-0.002429

zip	0.073175	0.014159	-0.018511	-0.012477	0.004486
customerAttr_a	0.000306	-0.008071	-0.004262	-0.017805	-0.027808
field_a	0.023651	0.010559	0.028043	0.057747	-0.008745
field_b	0.004318	-0.009476	0.020034	-0.005449	-0.012880
hour_b	0.006489	-0.007365	-0.016498	-0.013151	-0.002922
flag_a	0.007717	0.027672	-0.010917	-0.004212	0.105650
total	-0.029256	-0.019238	0.110985	0.016743	-0.020267
field_c	0.138208	-0.008878	-0.058493	-0.022408	-0.040561
field_d	0.002588	-0.021262	0.045699	0.045575	-0.084106
indicator_a	1.000000	-0.046539	0.002672	0.003341	-0.004515
indicator_b	-0.046539	1.000000	-0.004739	-0.016194	0.015896
flag_b	0.002672	-0.004739	1.000000	0.444027	-0.018151
flag_c	0.003341	-0.016194	0.444027	1.000000	-0.006936
flag_d	-0.004515	0.015896	-0.018151	-0.006936	1.000000
flag_e	0.011723	0.052316	0.067800	0.095568	-0.009132

	flag_e
amount	-0.140758
hour_a	-0.027809
zip	0.018130
customerAttr_a	0.011753
field_a	0.123510
field_b	0.018027
hour_b	-0.027886

flag_a	-0.037609
total	-0.140758
field_c	-0.008101
field_d	-0.037919
indicator_a	0.011723
indicator_b	0.052316
flag_b	0.067800
flag_c	0.095568
flag_d	-0.009132
flag_e	1.000000

Process finished with exit code 0

ملاحظه می شود که corr بین hour_a و hour_b تقریبا نزدیک به یک است و این یعنی که میتوانیم یکی از داده ها را حذف کنیم.

هم چنین کوریلیشن بین دو ویژگی total و amount نیز برابر یک است که میتوانیم یکی از این ویژگی ها را حذف کنیم.

ملاحظه میشود که به غیر از این دو ویژگی بقیه ی ویژگی ها خیلی به هم مرتبط نیستند بنابراین از بین این دو ویژگی ها میتوانیم یکی را حذف کنیم.

هم چنین بعضی از این فیلد ها نبز دارند به بک چیز مشخص اشاره میکنند برای مثال منظور از zip همان zip کد است و منظور از state ایالتی است که تراکنش در آن انجام شده و منظور از amount میزان تراکنش است .

هم چنین در این قسمت تعداد داده هایی با برچسب froud را با داده هایی با برچسب تراکنش سالم مقایسه میکنیم تا درک درستی از تعادل داد ها به دست آوریم .

با قطعه کد زیر

```
count_classes = pd.value_counts(label['fraud'], sort =
True).sort_index()
```

تعداد را در هر کلاس مقایسه می کنیم که داده های زیر به دست می آید.

[rows x 1 columns ۱۰۰۰۰۰]

۰ ۹۷۳۴۶

۱ ۲۶۵۴

که مشخص است که تعداد دسته های مثبت خیلی کمتر از منفی است .

علاوه بر این با مقایسه مشخص میشود که هر کدام از ستون ها دو تا ساعت در آن وجود دارد این یعنی احتمالا هر تراکنش دارد به دو کار اشاره میکند

همچنین مقدار بعضی از تراکنش ها برابر صفر است که این یعنی در آن ها پولی جابجا نشده است و همچنین هیچ تراکنشی با مقدار بیشتز از پنجاه دلار وجود ندارد.

اما نکته جالبی که در مورد تراکنش ها با مقدار صفر وجود دارد این است که flag_e بسیاری از آنها مقادیر بزرگ است و همچنین تعداد زیادی از ان ها دارای برچسب تقلب میباشند که این موضوع که این موضوع نشان دهنده این است که این ویژگی یک ویژگی بسیار مهم است در واقع کلن ۲۶۰۰ تا تقلب داریم و هزار و ششصد تا داده صفر اما تقریبا چهل و پنج درصد آن ها تقلب است که این موضوع جالبی است

هم چنین zip code و ایالت نیز با هم رابطه دارند

قسمت دو :

در زمان هایی که داده های ما نامتعادل هستند و برچسب یکی از داده ها خیلی بیشتر از بقیه است مانند این مثال ، راه هایی برای غلبه بر این مشکل وجود دارد که دو تا از معروف ترین راه ها over sampling و under sampling است .

راه های مختلف را برای این کار بررسی میکنیم.

بهترین راهی که وجوددارد جمع آوری داده های بیشتر است که برای این مساله عملی نیست .

اما راه دیگر که کاربردی است resample کردن داده ها به صورتی است که توزیع داده ها به صورت پنجاه پنجاه برابر شود که برای این کار از دو تکنیک under sampling و over sampling استفاده میشود .

Under sampling یعنی این که از داده هایی که برچسب ان ها تعداد بیشتری در دیتاست ما وجود دارد حذف کنیم که تعداد داده های مثبت و منفی به تعادل برسد

Over sampling یعنی چند بار از داده ها با برچسب کمتر استفاده کنیم و آن را کپی کنیم که دیتاست به تعادل برسد.

ما در این مساله ابتدا از روش under sampling استفاده می کنیم .

به این صورت که به تعداد داده های مثبت از داده های منفی جدا می کنیم و دیتاست جدید را میسازیم.

```
number_records_fraud = len(data[data.fraud == 1])
fraud_indices = np.array(data[data.fraud == 1].index)
normal_indices = data[data.fraud == 0].index
#print(fraud_indices)
random_normal_indices = np.random.choice(normal_indices,
number_records_fraud, replace = False)
random_normal_indices = np.array(random_normal_indices)

#dar do khatte bala tedede barabar ba dade haye mosbat ra az
dade haue manfi jada mikonim

under_sample_indices =
np.concatenate([fraud_indices,random_normal_indices])
under_sample_data = data.iloc[under_sample_indices,:]


X_undersample = under_sample_data.ix[:,
under_sample_data.columns != 'fraud']
```

```
y_undersample = under_sample_data.ix[:,
under_sample_data.columns == 'fraud']
```

سپس از اور سمپلینگ استفاده میکنیم و داده های تخلف را چند بار کپی میکنیم اما چون نتجه آندرسمپلینگ بهتر است نتایج آن را در گزارش می آوریم

قسمت سوم :

برای مقایسه مدل ها از سه معیار accuracy و f1_score و recall استفاده میکنیم اما معیار بسیار مهم برای ما recall است زیرا هدف تشخیص درست تراکنش های تقلب است اما نکته ای که در این زمینه باید در نظر گرفت این است که اگر ما همه را تقلب تشخیص بدهیم هم recall برای ما برابر یک میشود بنابراین استفاده از recall خالی برای ما خوب نیست به همین منظور از معیار f1-score نیز استفاده میکنیم زیرا در واقع این معیار ترکیبی از دو معیار ریکال و بریسیژن را به ما میدهد بنابراین معیار خوبی است هر چند بازهم تا حدودی سلیقه ای است و باید دید که هدف طرف مقابل که خواستار انجام این کار است چیست اما با در این گزارش از دو معیار f1 و recall برای مقایسه استفاده میکنیم .

هم چنین با استفاده از under sampling از داده مورد نظر را به تعادل میرسانیم و از آنجایی که داده تست شده مجموعه متعادلی از داده هاخواهد بود ما نیز معیارهایمان را بر روی مجموعه متعادلی از داده ها تست میکنیم .

ابتدا روش logistic regression :

با اعداد مختلفی با استفاده از for این مدل را امتحان می کنیم که با اعداد زیر نتیجه بهتری به دست می آید.

```
lr = LogisticRegression(C =20, penalty = 'l1')
lr.fit(X_train_undersample,y_train_undersample.values.ravel())
scores = cross_val_score(lr, X_undersample,
y_undersample.values.ravel(), cv = 10, scoring='recall')
```

برای این مدل

recall= 0.687466307278

precision= 0.728477743872

f1_score= 0.696706091111

acuuracy= 0.714582210243

حال با استفاده از درخت تصمیم مساله را حل میکنیم که اعداد زیر بهترین اعداد با استفاده از for به دست آمده اند .

```
dt = tree.DecisionTreeClassifier(random_state=1, max_depth=20)
dt = dt.fit( X_undersample, y_undersample)
scores = cross_val_score(dt, X_undersample,
y_undersample.values.ravel(), cv = 10, scoring='f1')
```

recall= 0.703467158462

precision= 0.706296340734

f1_score= 0.702090374147

acuuracy= 0.706669031068

حال با استفاده از Random forest مساله را حل میکنیم که به اعداد زیر می رسیم

```
rf = RandomForestClassifier(n_estimators = 100, warm_start=True,
random_state=2, min_samples_leaf=2,max_depth=21)
rf =
rf.fit(X_train_undersample,y_train_undersample.values.ravel())
scores = cross_val_score(rf, X_undersample,
y_undersample.values.ravel(), cv = 10, scoring='accuracy')
```

recall= 0.68463895588

precision= 0.819546159876

f1_score= 0.73683416345

acuuracy= 0.761126400908

و در نهایت با استفاده از شبکه عصبی به اعداد زیر میرسیم .

قسمت چهار :

با استفاده از ترکیب مدل های مختلف مدل جدیدی میسازیم و جواب را بر اساس آن امتحان میکنیم و به اعداد زیر میرسیم .

```
lr = LogisticRegression(C =20, penalty = 'l1')
dt = tree.DecisionTreeClassifier(random_state=1, max_depth=20)
rf = RandomForestClassifier(n_estimators = 100, warm_start=True,
random_state=2, min_samples_leaf=2,max_depth=21)
combine=VotingClassifier(estimators=[("lr",lr),("dt",dt),("rg",r
f)],voting='soft')
combine.fit( X_undersample, y_undersample)
scores = cross_val_score(combine, X_undersample,
y_undersample.values.ravel(), cv = 10, scoring='recall')
```

recall= 0.7121378919

precision= 0.751765980442

f1_score= 0.721140296

acuuracy= 0.719302028657

قسمت پنج :

حال ویژگی جدیدی به این داده ها اضافه میکنیم و آن را exp می نامیم که در واقع یک flag است که نشان می دهد که آیا مقدار پول جابجا شده درتراکنش از یک حدی بیشتر است یا خیر.این آستانه را برابر با مقدار صفر در نظر میگیریم در واقع این ویژگی یک فلگ است که داریم به داده مان اضافه میکنیم و ملاحظه میشودکه با اضافه کردن این ویژگی ملاحظه میشود که مقدار fscore و recall بهبود می یابد در حدود سه صدم

سوال ۲

قسمت یک :

سوال دو :

ابتدا از classifier درخت تصمیم استفاده می کنیم .

ابتدا عمل پیش پردازش را انجام میدهیم و پس از پر کردن missing values درخت تصمیمان را با اعداد مقابل میسازیم

```
max_depth = 10
min_samples_split =5
my_tree=tree.DecisionTreeClassifier(max_depth=max_depth,min
_samples_split=min_samples_split,random_state=1)
my_tree.fit(features,target)
my_prediction = my_tree.predict(test_features)
print(my_prediction)
```

ملاحظه میشود که با آپلود فایل به دقت ۷۶ درصد میرسیم .

**فایل ساخته شده با نام mysolution1_sex.csv در فایل های پروژه موجود است.

حال از classifier نزدیک ترین همسایه های مشترک یا knn برای انجام پیش بینی استفاده می کنیم .

و k را برابر بایک در نظر میگیریم و مشاهده میشود که جوابی که به دست می آید برابر ۶۳ درصد میباشد

```
features = train[["Pclass", "Sex", "Age", "Fare","SibSp",
"Parch", "Embarked"]].values

test_features = test[["Pclass", "Sex", "Age",
"Fare","SibSp", "Parch", "Embarked"]].values
```

```
target = train["Survived"].values

neigh=KNeighborsClassifier(n_neighbors=1)

neigh.fit(features,target)

my_prediction=neigh.predict(test_features)
```

حال از naïve bayes classifier برای پیش بینی استفاده می کنیم .

```
clf=GaussianNB()
clf.fit(features,target)

my_prediction=clf.predict(test_features)
```

و جواب را به عنوان mysolution_sex ذخیره میکنیم و به دقت ۷۵ میرسیم

قسمت دوم :

در این قسمت با استفاده از ترکیب مدل های بالا یک مدل جدید میساریم و هر کدام که تعداد بیشتری به آن برچسپ رای دادند آن را به عنوان جواب در نظز میگیریم

```
clf=GaussianNB()
neigh=KNeighborsClassifier(n_neighbors=1)
my_tree=tree.DecisionTreeClassifier(max_depth=max_depth,min_samp
les_split=min_samples_split,random_state=1)



combine=VotingClassifier(estimators=[("gnb",clf),("knn",neigh),(
"dec",my_tree)],voting='soft')
combine.fit(features,target)
my_prediction=combine.predict(test_features)
```

قسمت سوم :

در این قسمت ویژگی childرا اضافه میکنیم به این صورت که اگر شخصی بالای هجده سال سن داشت یعنی این که این شخص بچه است و اگر بالای هجده سال نبود بزرگسال یه حساب می آید همچنین میتوان ویژگی تعداد اعضای خانواده را نیز اضافه کرد اما افزودن این ویژگی ها خیلی تاثیری در جوای نهایی ندارد .

```
train["Child"][train["Age"] < 18] = 1
train["Child"][train["Age"] >= 18] = 0
```

```
train_two["family_size"]
=train_two["SibSp"]+train_two["Parch"]+1
test_two=ini.test.copy()
test_two["family_size"] =test_two["SibSp"]+test_two["Parch"]+1
```