به‌نام خداوند بخشنده و مهربان

تمرین چهارم: لاجستیک رگرسیون
استاد: محمدعلی نعمت‌بخش

درس: تحلیل سیستم داده‌های حجیم
دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام‌خانوادگی: عطیه نیکبخت

آدرس گیت: https://github.com/AtiyehNikbakht/LogisticRegression.git

- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و همراه نوتبوک تمرین در کوئرا ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی تمرین را در سند خود بارگذاری کنید.**

در این تمرین هدف کار با کتاب‌خانه‌ی pyspark و همچنین کتاب‌خانه‌ی یادگیری ماشین آن است.

برای این منظور دیتاستی در اختیار شما قرار گرفته است. اطلاعات کاربران شرکتی در اختیار شما قرار داده شده است. این شرکت اطلاعات چند ماه از کاربرانش را برچسب گذاری کرده است. این برچسب به معنای این است که آیا مشتری شرکت را ترک کرده و دیگر از خدمات آن استفاده می‌کند یا خیر. انتظار می‌رود با بررسی دقیق مجموعه‌ی داده و تحلیل داده‌گان آن در نهایت مدل پیشبینی کننده‌ای برای این شرکت طراحی کنید.

هر یک از موارد زیر را به دقت بررسی کنید و نتایج آن را در قالب اسکرین شات و تحیل خود در سند ذکر کنید.

- **قدم اول:** دیتاست داده شده را پیش پردازش کنید. مقادیر NA را مقدار دهی کنید تحلیل داده اکتشافی (EDA) را به خوبی انجام دهید. این ستونها براساس ماهیت خود میتواند تولید کننده ویژگیهای بیشتری باشند که ممکن است دقت مدل شما را بالاتر ببرند. در این مرحله همبستگی و ارتباط بین تمام ویژگی هایی که میتوانید استخراج کنید را بررسی کنید. (نمودارهای لازم برای تحلیل دادگان ترسیم شود.)
- **قدم دوم:** عملیات feature engineering را به خوبی برای داده‌گان خود انجام دهید و دلیل انتخاب هریک از ستون‌ها یا عدم انتخاب آن‌ها را به صورت منطقی بیان کنید. (با نمودار و تحلیل آن، با کمک EDA انجام شده)
- **قدم سوم:** الگوریتم Logestic Regression را بر روی داده‌های خود اعمال کنید.

- **قدم چهارم:** دقت مدل خود را ارزیابی کنید. (در این مرحله شـــما باید مراحل آزمایش، تعداد دادگان ترین و تست، احتمال صحیح بودن یک برچسب که مدل پیشبینی کرده است، را تعیین کنید)
- نتایج مدل قبل و بعد از پیش‌پردازش را مقایسه کنید.

نکات مهم

- برای پیش‌پردازش دادگان و یادگیری ماشین فقط از کتابخانه پای‌اسپارک استفاده شود.(pandas مجاز نیست).
- نمودارهای ترسیمی حتما همراه با تحلیل در سند آورده شوند.
- کپی نکنید! از قبل تمام کدهای نوشته شده در اینترنت جمع‌اوری شده است کپی کردن شما مشخص می‌شود.

ابتدا کتابخانه‌های مورد نیاز را داخل پروژه وارد می‌کنیم که عبارتند از کتابخانه pyspark، matplotlib، numpy و seaborn. سپس فایل داده را خوانده و در df می‌ریزیم. پیش‌پردازش روی داده‌های دیتافریم df انجام می‌شود و در همین دیتافریم ذخیره می‌شود. دیتافریم dfWop جهت بررسی مدل Logistic regression بدون پیش‌پردازشی روی داده‌ها انجام می‌شود.

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import col,isnan, when, count
from pyspark.sql.types import FloatType
import numpy as np
import matplotlib.pyplot as plt
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml.stat import ChiSquareTest
import seaborn as sns; sns.set_theme()
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

spark = SparkSession.builder.appName("Dataframe_practice").getOrCreate()
df = spark.read.csv('data.csv', header=True)
dfWop = df
```

با استفاده از تابع isNull() می‌توان به داده‌های ناموجود دسترسی داشت. در شکل زیر تعداد مقدارهای ناموجود هر ستون را مشاهده می‌کنید.

```
for c in df.columns:
     print(df.select(count(when(col(c).isNull(), c)).alias(c)).collect()[0])
```

```
Row(customerID=253)
Row(gender=235)
Row(SeniorCitizen=266)
Row(Partner=225)
Row(Dependents=242)
Row(tenure=225)
Row(PhoneService=269)
Row(MultipleLines=263)
Row(InternetService=230)
Row(OnlineSecurity=230)
Row(OnlineBackup=243)
Row(DeviceProtection=254)
Row(TechSupport=264)
Row(StreamingTV=249)
Row(StreamingMovies=219)
Row(Contract=230)
Row(PaperlessBilling=257)
Row(PaymentMethod=246)
Row(MonthlyCharges=243)
Row(TotalCharges=251)
Row(Label=208)
```

جهت انجام پیش‌پردازش ابتدا سطرهایی که ستون Label آن‌ها مقداری ندارد را حذف می‌کنیم. سپس ستون customerID را حذف می‌کنیم زیرا هر مشتری شناسه مخصوص به‌خود را دارد و این ستون تأثیری بر Label ندارد. سپس اگر سطری تمام مقادیر ستون‌هایش برابر با None باشد را حذف می‌کنیم. در ادامه سطرهایی که تمام مقادیر ستون‌هایشان باهم برابر است و تکراری هستند را حذف می‌کنیم.

جهت پر کردن بقیه مقادیر ناموجود براساس نوع ستون عمل می‌کنیم. ابتدا مقادیر ناموجود ستون‌هایی که عددی نیستند و اسمی (Categorical) هستند را با بیشترین مقداری که تکرار شده، جایگزین می‌کنیم. ستون SeniorCitizen به دلیل اینکه دارای مقادیری مانند ۰ و ۱ است از دستور دیگری جهت پیدا کردن بیشترین مقدار آن استفاده می‌کنیم و در ادامه نیز مشاهده می‌کنیم که این ستون دارای مقادیر پرت می‌باشد و آن‌ها را نیز با مقدار مناسب جایگزین می‌کنیم. ستون‌های عدد باقی‌مانده را با مقدار میانگین آن‌ها جایگزین می‌کنیم. در نهایت دوباره سطرهای تکراری را حذف می‌کنیم زیرا وجود داده‌های تکراری ممکن است منجر به overfit شدن مدل شود. برای دیتافریم dfWop هر سطری که دارای مقادیر ناموجود باشد را حذف می‌کنیم.

```
df = df.filter(~(col('label').isNull()))
df = df.drop('customerID')
df = df.na.drop("all")
df = df.dropDuplicates()
dfWop = dfWop.na.drop("any")

for i in df.columns:
    if i not in ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']:
        df = df.fillna(value=df.agg(F.max(i)).collect()[0][0], subset=i)

    elif i in ['tenure', 'MonthlyCharges', 'TotalCharges']:
        val = round(df.agg(F.avg(i)).collect()[0][0], 1)
        df = df.na.fill(value=str(val), subset=i)
        df = df.withColumn(i, df[i].cast(FloatType()))
        dfWop = dfWop.withColumn(i, dfWop[i].cast(FloatType()))

    elif i == 'SeniorCitizen':
        df = df.fillna(value=df.groupby('SeniorCitizen').count().sort('count').collect()[-1][0], subset=i)

df = df.dropDuplicates()
```

همان‌طور که در شکل زیر مشاهده می‌کنید تمام مقادیر ناموجود دیتا فریم df مقداردهی شدند.

```
for c in df.columns:
     print(df.select(count(when(col(c).isNull(), c)).alias(c)).collect()[0])
```

```
Row(gender=0)
Row(SeniorCitizen=0)
Row(Partner=0)
Row(Dependents=0)
Row(tenure=0)
Row(PhoneService=0)
Row(MultipleLines=0)
Row(InternetService=0)
Row(OnlineSecurity=0)
Row(OnlineBackup=0)
Row(DeviceProtection=0)
Row(TechSupport=0)
Row(StreamingTV=0)
Row(StreamingMovies=0)
Row(Contract=0)
Row(PaperlessBilling=0)
Row(PaymentMethod=0)
Row(MonthlyCharges=0)
Row(TotalCharges=0)
Row(Label=0)
```

برای دیتافریم dfWop نیز تمام مقادیر ناموجود حذف شده‌اند.

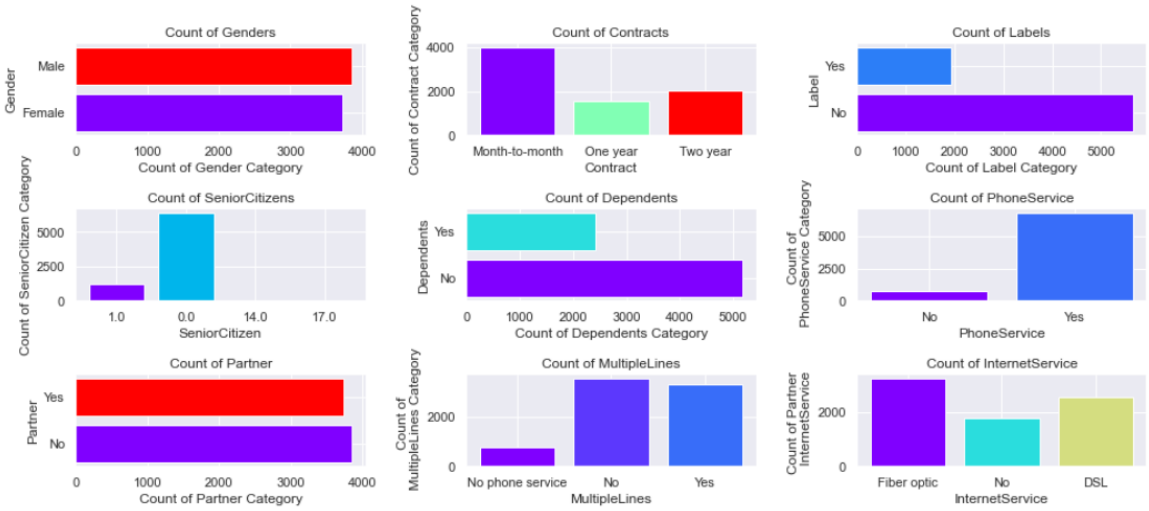
```
for c in dfWop.columns:
     print(dfWop.select(count(when(col(c).isNull(), c)).alias(c)).collect()[0])
```

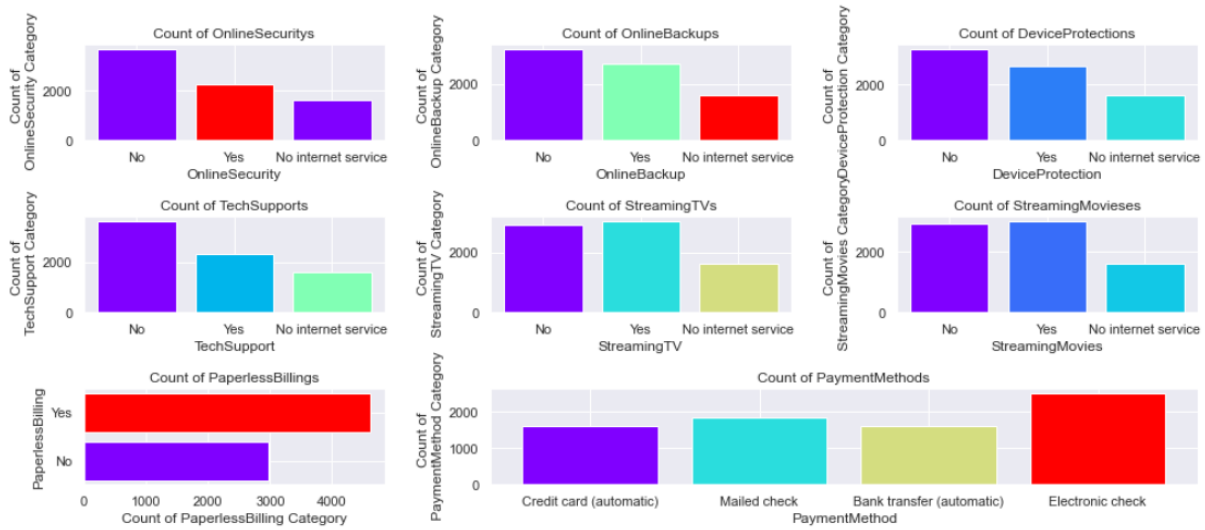
```
Row(customerID=0)
Row(gender=0)
Row(SeniorCitizen=0)
Row(Partner=0)
Row(Dependents=0)
Row(tenure=0)
Row(PhoneService=0)
Row(MultipleLines=0)
Row(InternetService=0)
Row(OnlineSecurity=0)
Row(OnlineBackup=0)
Row(DeviceProtection=0)
Row(TechSupport=0)
Row(StreamingTV=0)
Row(StreamingMovies=0)
Row(Contract=0)
Row(PaperlessBilling=0)
Row(PaymentMethod=0)
Row(MonthlyCharges=0)
Row(TotalCharges=0)
Row(Label=0)
```

در ادامه تمام ستون ها را دسته بندی کرده و با استفاده از دستورات زیر برای هر داده Categorical یک نمودار رسم می‌کنیم تا دسته‌ها و تعداد آن‌ها را بهتر مشاهده کنیم.

```
#Count of Gender
itemList = {}
for i in df.groupby('gender').count().collect():
    itemList[i[0]] = i[1]

fig = plt.figure(figsize=(15, 6))
plt.subplot(3,3,1)
color = plt.cm.rainbow(np.linspace(0, 1, 2))
plt.xlabel('Count of Gender Category')
plt.ylabel('Gender')
plt.title('Count of Genders')
plt.barh(np.array(list(itemList.keys())), np.array(list(itemList.values())), align='center', color=color)


#Count of Contract
itemList = {}
for i in df.groupby('Contract').count().collect():
    itemList[i[0]] = i[1]

plt.subplot(3,3,2)
color = plt.cm.rainbow(np.linspace(0, 1, 3))
plt.ylabel('Count of Contract Category')
plt.xlabel('Contract')
plt.title('Count of Contracts')
plt.bar(np.array(list(itemList.keys())), np.array(list(itemList.values())), align='center', color=color)


#Count of Label
itemList = {}
for i in df.groupby('Label').count().collect():
    itemList[i[0]] = i[1]

plt.subplot(3,3,3)
color = plt.cm.rainbow(np.linspace(0, 1, 7))
plt.xlabel('Count of Label Category')
```

در شکل زیر نمودارها نمایش داده شده‌اند. همانطور که مشاهده می‌کنید ستون SeniorCitizen دارای مقادیر پرت ۱۷ و ۱۴ است. در ادامه با توجه به تعداد این مقادیر با آن‌ها برخورد مناسبی می‌شود.





باتوجه به تعداد مقادیر پرت ستون SeniorCitizen می‌توان از این سطرها چشم‌پوشی کرد. بنابراین آن‌ها را حذف می‌کنیم.

```
print(df.groupby('SeniorCitizen').count().sort('count').collect())
df = df.filter('SeniorCitizen < 14')
print(df.groupby('SeniorCitizen').count().sort('count').collect())
```

```
[Row(SeniorCitizen='17.0', count=1), Row(SeniorCitizen='14.0', count=8), Row(SeniorCitizen='1.0', count=1217), Row(SeniorCitize
n='0.0', count=6361)]
[Row(SeniorCitizen='1.0', count=1217), Row(SeniorCitizen='0.0', count=6361)]
```

در مرحله بعد به سراغ محاسبه همبستگی بین متغیرها می‌رویم. می‌توان برای داده‌های اسمی از آزمون Chisquare استفاده کرد و یا از Pearson correlation برای داده‌های عددی استفاده می‌کنیم. ابتدا دو ستون Label و gender را با استفاده از آزمون Chisquare بررسی می‌کنیم. در شکل زیر نحوه محاسبه آورده شده است. برای استفاده از این آزمون مقدار ویژگی‌ها و برچسب باید از نوع Categorical باشند. پس از محاسبه مقدار باید به جدول توزیع آزمون Chisquare مراجعه کرده و بسته به درجه آزادی و مقدار به‌دست آمده، همبستگی بین دو ستون را مشاهده کرد و یا با استفاده از دستورات زیر می‌توان مقدار را پیدا کرد.

```
genInd = StringIndexer(inputCol="gender", outputCol="genInd")
genInd = genInd.fit(df).transform(df)
genOHE = OneHotEncoder(inputCol='genInd',outputCol='genOHE')
genOHE = genOHE.fit(genInd).transform(genInd)
genVec = VectorAssembler(inputCols=['genOHE'], outputCol='genVec')
genVec = genVec.transform(genOHE)

labInd = StringIndexer(inputCol="Label", outputCol="labInd")
labInd = labInd.fit(df).transform(df)
lab = [i[0] for i in labInd.select('labInd').collect()]
gen = [i[0] for i in genVec.select('genVec').collect()]

data = list(zip(lab, gen))
df1 = spark.createDataFrame(data, ['label', 'gender'])
r = ChiSquareTest.test(df1, 'gender', 'label').head()

print("pValues: " + str(r.pValues))
print("degreesOfFreedom: " + str(r.degreesOfFreedom))
print("statistics: " + str(r.statistics))
```

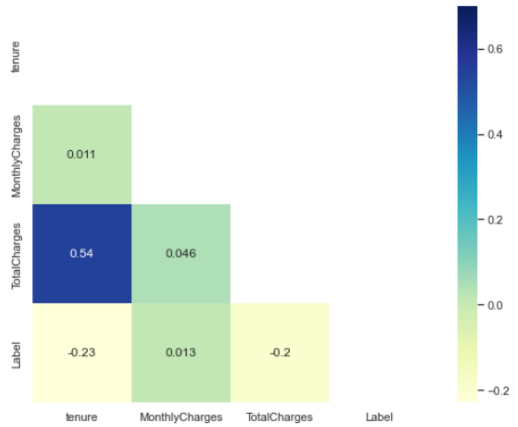
```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packag
3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(

pValues: [0.05729946085420601]
degreesOfFreedom: [1]
statistics: [3.6138750711807086]
```

در ادامه به‌سراغ محاسبه Pearson correlation بین تمام ویژگی‌ها می‌رویم. ابتدا ویژگی‌های Categorical را با استفاده از StringIndexer به ویژگی‌های عددی تبدیل می‌کنیم. سپس برروی این ویژگی‌ها OneHotEncoder را (جهت نگاشت ستونی از شاخص‌های دسته‌بندی به بردارهای دودویی) انجام می‌دهیم و در نهایت از ویژگی‌های به‌دست آمده و جهت ادغام ویژگی‌های عددی یک بردار با استفاده از VectorAssembler می‌سازیم. این بردار در ستون features قرار می‌گیرد (در اینجا نام ستون خروجی features است) البته باید توجه کرد که Label همراه با بقیه‌ی ویژگی‌ها به یک بردار تبدیل نشود زیرا از خروجی این بردار برای آموزش مدل استفاده می‌شود. بنابراین ستون Label مانند بقیه ویژگی‌ها از StringIndexer و OneHotEncoder استفاده می‌کند تا به ویژگی عددی تبدیل شود اما با بقیه در یک بردار قرار نمی‌گیرد. جهت استفاده از StringIndexer و OneHotEncoder باید ابتدا ستون‌های ورودی و خروجی را مشخص کنیم و سپس برروی داده‌هایی که می‌خواهیم fit کرده و در نهایت توسط تابع transform تغییر را انجام دهیم.

```
colInd = []
colistr = []
colOHE = []
colInt = []
for colm,types in df.dtypes:
    if types == 'string':
        colistr.append(colm)
        colInd.append(colm + 'Ind')
        colOHE.append(colm + 'OHE')
    else:
        colInt.append(colm)

#Convert Categorical data to Numerical data
sindex = StringIndexer(inputCols=colistr, outputCols=colInd)
Ohe = OneHotEncoder(inputCols=colInd, outputCols=colOHE)

#Creating a vector of all attributes(without Label)
colOHE.remove('LabelOHE')
colList = colOHE + colInt
colVec = VectorAssembler(inputCols=colList, outputCol='features')

ind = sindex.fit(df).transform(df)
onehot = Ohe.fit(ind).transform(ind)
vec = colVec.transform(onehot)
```

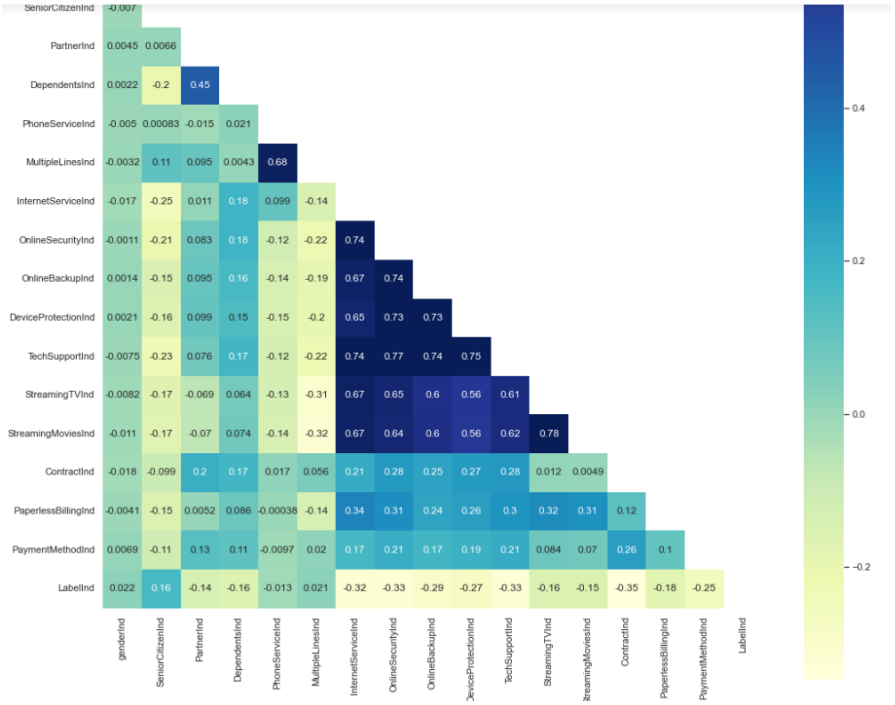
جهت محاسبه دیگر همبستگی‌ها از دستورات زیر استفاده می‌کنیم. همانطور که در شکل زیر مشاهده می‌کنید ابتدا همبستگی بین ویژگی‌هایی که از ابتدا عددی بوده‌اند و برچسب محاسبه شده است. هم‌چنین میزان همبستگی هر ویژگی با ویژگی غیر برچسب نیز محاسبه شده است.

```
myList = []
for i,j,k,m in zip(df.select('tenure').collect(), df.select('MonthlyCharges').collect(), df.select('TotalCharges').collect(), vec
    myList.append((i[0], j[0], k[0], m[0]))
dfCorr = spark.createDataFrame(myList, ['tenure','MonthlyCharges', 'TotalCharges', 'Label'])
corrmat = dfCorr.toPandas().corr()

mask = np.zeros_like(corrmat)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(10, 7))
    ax = sns.heatmap(corrmat, mask=mask, vmax=0.7, square=True, cmap="YlGnBu", annot=True)
```



در ادامه نیز میزان همبستگی میان ویژگی‌هایی که در ابتدا Categorical بوده‌اند و برچسب بررسی شده است. هم‌چنین میزان همبستگی هر ویژگی با ویژگی غیر برچسب نیز محاسبه شده است.

```
corrmat = vec.select(['genderInd', 'SeniorCitizenInd', 'PartnerInd', 'DependentsInd', 'PhoneServiceInd', 'MultipleLinesInd', 'Int
mask = np.zeros_like(corrmat)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(17, 17))
    ax = sns.heatmap(corrmat, mask=mask, vmax=0.7, square=True, cmap="YlGnBu", annot=True)
```



همانطور که در شکل بالا مشاهده می‌کنید بین برخی از ویژگی‌ها میزان همبستگی بالایی وجود دارد. می‌توان برخی از آن‌ها را حذف نمود. اما در صورت حذف یکی از ویژگی‌ها دقت مدل کاهش پیدا می‌کند. بنابراین در اینجا هیچکدام از ویژگی‌ها را حذف نمی‌کنیم.

در این مرحله مدل خود را باتوجه به داده‌هایی که پیش‌پردازش شده‌اند آموزش داده و سپس با داده‌های تست، دقت مدل را ارزیابی می‌کنیم. مقدار seed را روی ۲۵۰ قرار می‌دهیم. دقت مدل باتوجه به موارد ذکرشده برابر با ۰.۷۳ است.

```
#Training Logistic Regression Model with Preprocessing Data
trainDF, testDF = vec.select(['features', 'LabelInd']).randomSplit([.8, .2], seed=250)

logReg = LogisticRegression(featuresCol='features',labelCol='LabelInd')
fitModel = logReg.fit(trainDF)
results = fitModel.transform(testDF)

myEval = BinaryClassificationEvaluator(rawPredictionCol='prediction', labelCol='LabelInd')
AUC = myEval.evaluate(results)
print("AUC score is : ",AUC)
```

```
AUC score is :  0.7349586288416076
```

حال دقت مدل را بدون انجام پیش‌پردازش برروی داده‌ها انجام می‌دهیم. همانطور که مشاهده می‌کنید دقت برابر با ۰.۶۰ است.

```
#Training Logistic Regression Model without Preprocessing Data
colInd = []
colistr = []
colOHE = []
colInt = []
for colm,types in dfWop.dtypes:
    if types == 'string':
        colistr.append(colm)
        colInd.append(colm + 'Ind')
        colOHE.append(colm + 'OHE')
    else:
        colInt.append(colm)

sWop = StringIndexer(inputCols=colistr, outputCols=colInd)
OheWop = OneHotEncoder(inputCols=colInd, outputCols=colOHE)
colInd.remove('LabelInd')
colList = colInd + colInt
colVecWop = VectorAssembler(inputCols=colList, outputCol='features')

indWop = sWop.fit(dfWop).transform(dfWop)
onehotWop = OheWop.fit(indWop).transform(indWop)
vecWop = colVecWop.transform(onehotWop)

trainDF, testDF = vecWop.select(['features', 'LabelInd']).randomSplit([.8, .2], seed=250)

logReg = LogisticRegression(featuresCol='features',labelCol='LabelInd')
fitModel = logReg.fit(trainDF)
results = fitModel.transform(testDF)

myEval = BinaryClassificationEvaluator(rawPredictionCol='prediction', labelCol='LabelInd')
AUC = myEval.evaluate(results)
print("AUC score is : ",AUC)
```

```
AUC score is :  0.6054141099844311
```