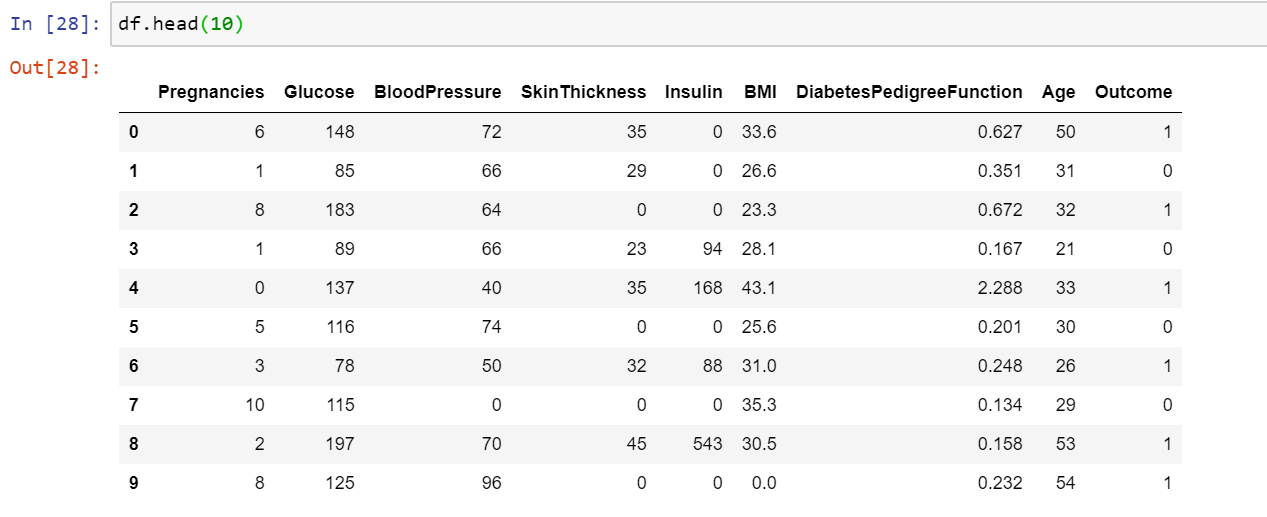
**گزارش تمرین سوم درس داده کاوی**

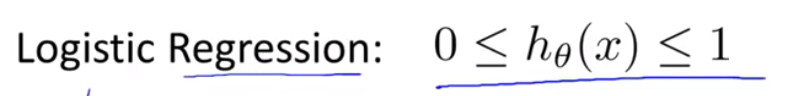
**فاطمه غلام زاده -9531060**

* **سوال اول :**

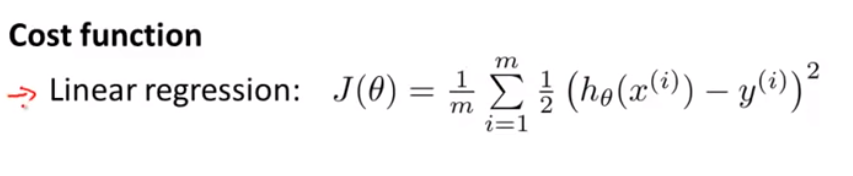
در این مسئله می خواهیم داده های مربوط به بیماری دیابت را با استفاده از رگرسیون خطی دسته بندی کنیم. برای اینکه یک دید کلی از دیتا و فیچر های آن بدست آوریم پس از لود کردن دیتاست ، با دستور df.head(10) ده سطر اول دیتا به همراه فیچر های آن را مشاهده می کنیم :



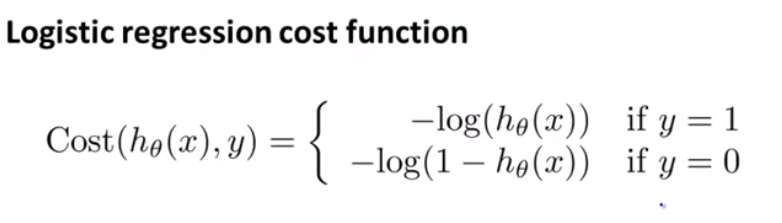
برای پیاده سازی این مسئله از روش logistic regression استفاده می کنیم. در این روش داریم :



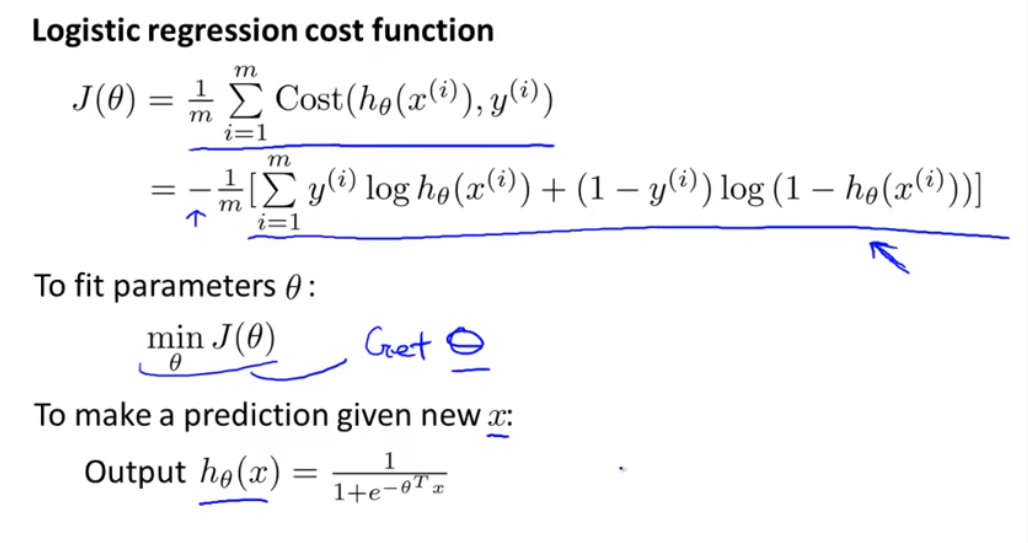
تفاوت رگرسیون خطی با logistic در تابع cost آن هاست. در رگرسیون خطی تابع cost به شکل زیر است :



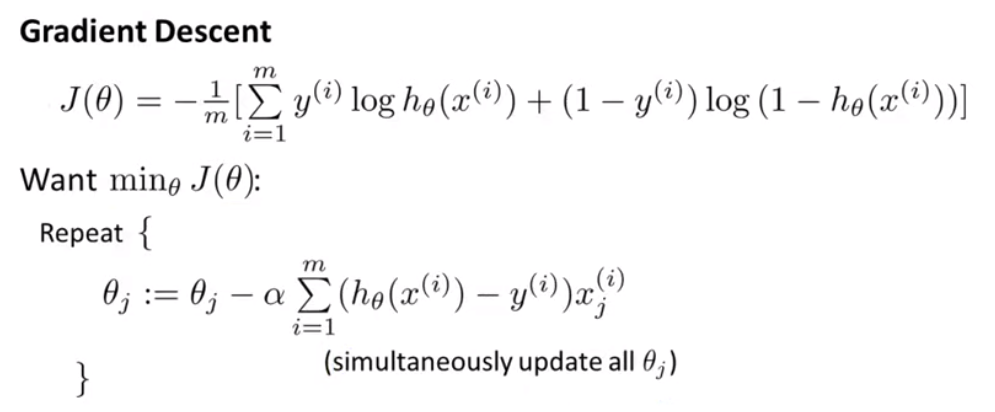
در رگرسیون logistic تابع cost به شکل زیر است :



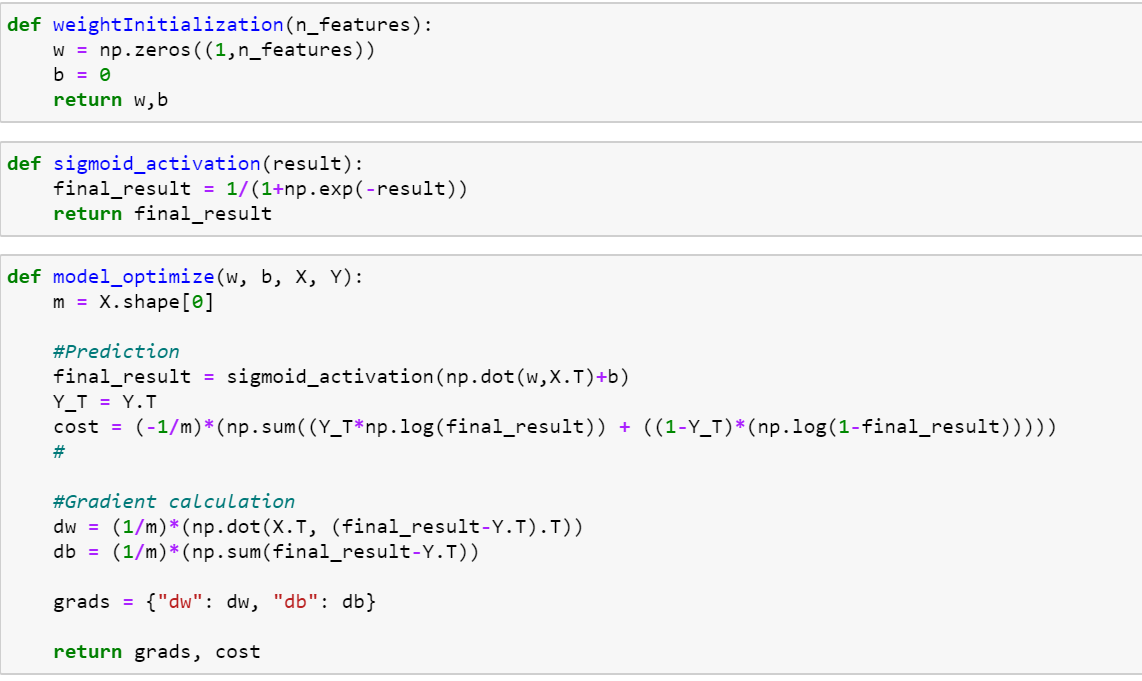
برای اینکه وقتی یک x جدید می بینم پیش بینی را انجام بدهیم به صورت زیر عمل می کنیم :

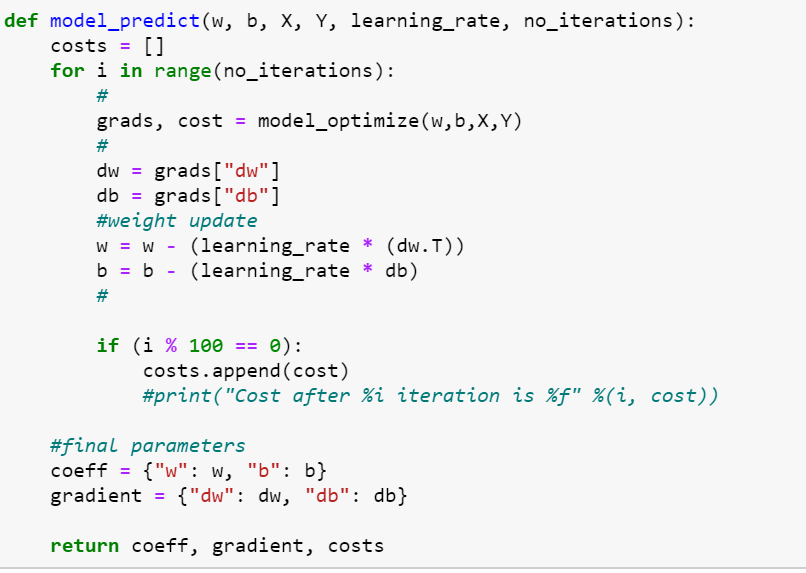


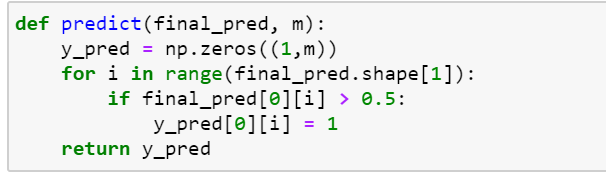
هم چنین از روش gradient descent استفاده می کنیم که به صورت زیر است :

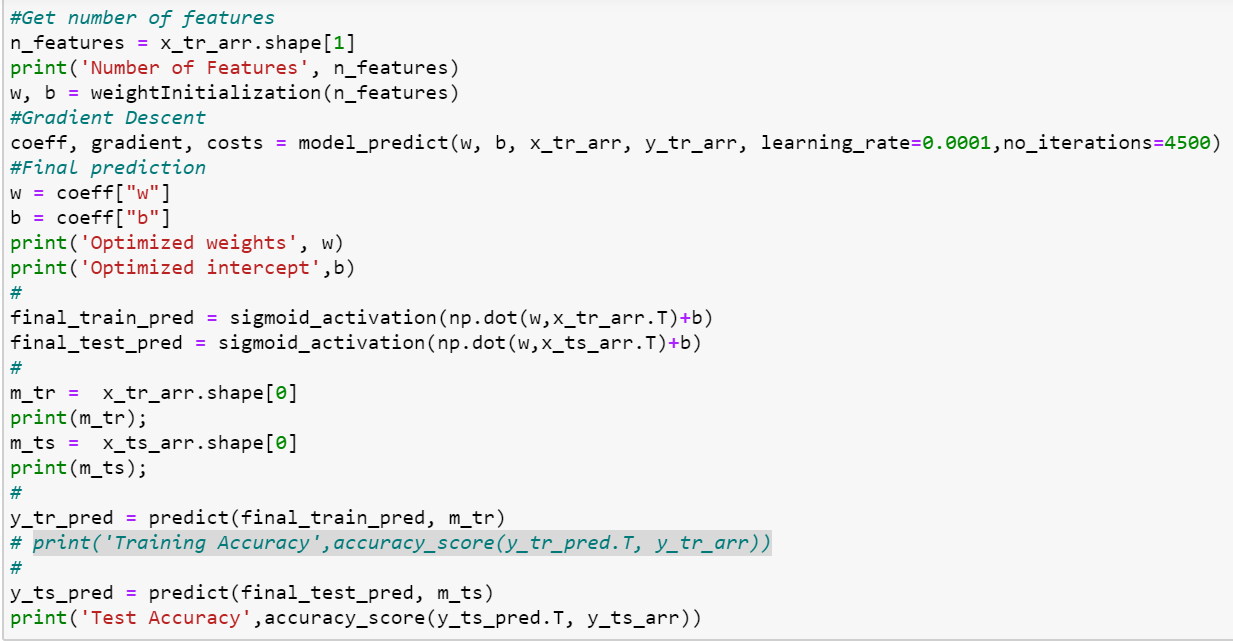


پیاده سازی این فرمول ها در کد به شکل زیر است :

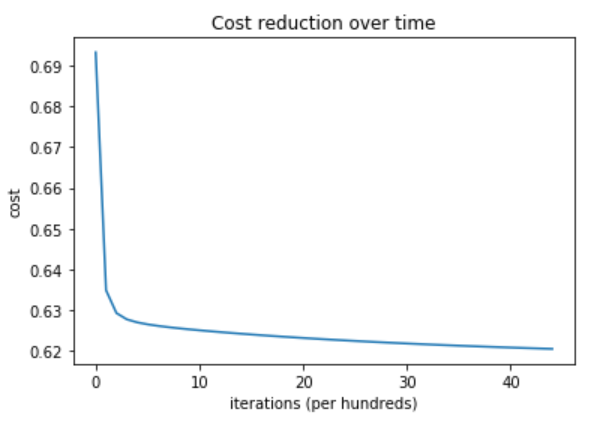
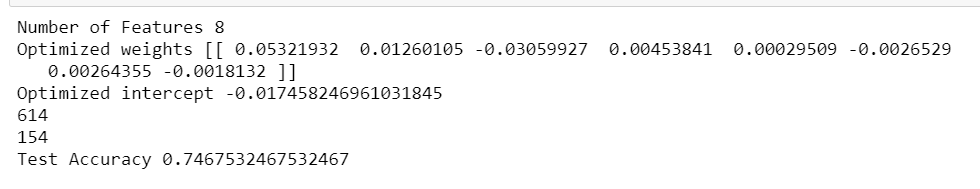








در نهایت دقت به دست آمده برای داده های تست برابر با 74 درصد می شود و نمودار تابع cost به شکل زیر خواهد بود :

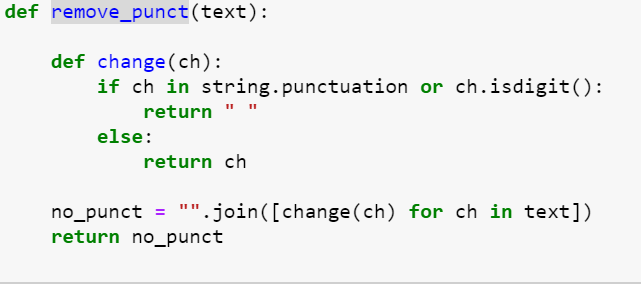
  


* **سوال دوم :**

در این سوال قصد داریم تا به کمک روش naïve bayes پیش بینی کنیم که یک سوال صادقانه است یا نه . بدین منظور پس از لود کردن دیتاست باید پیش پردازش هایی رو روی داده ها انجام بدهیم . از آن جایی که دیتاست داده شده شامل جملاتی پرسشی به زبان طبیعی است بنابراین لازم است تا پردازش های موجود برای زبان طبیعی را روی آن انجام بدهیم. از جمله این پردازش ها به صورت زیر است :

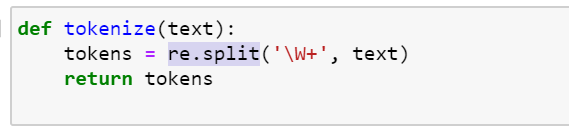
1. حذف علائم نگارشی و اعداد

بدین منظور تابع remove\_punct را تعریف می کنیم که با دریافت یک متن ، هر کاراکتر در متن را چک میکند ، اگر علائم نگارشی یا عدد بود آن را حذف می کند .



1. Tokenize کردن متن

تابع زیر را تعریف می کنیم که یک متن می گیرد و token های آن را بر میگرداند .



1. حذف stop word ها

یک فایل متنی به نام sw.txt داریم که شامل کلماتی نظیر شمایر ، افعال کمکی ، فعل های کوتاه و .... است. این فایل را در کد خود لود می کنیم .تابع زیر را تعریف کرده ایم که یک متن token شده می گیرد و اگر شامل کلماتی باشد که در این فایل هست ، آن ها را حذف میکند.



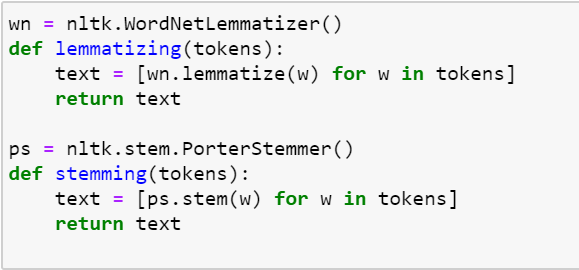
1. حذف کلمات کوتاه

یک فانکشن داریم که token ها را می گیرد و آن هایی که طول کمتر از 2 داشته باشند را حذف میکند.

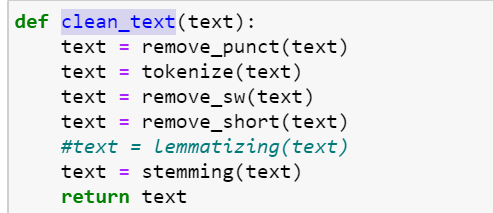


1. ریشه یابی کلمات

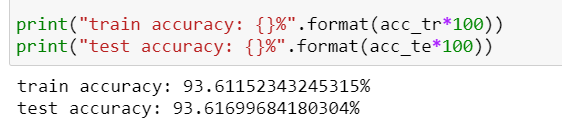
به کمک دو فانکشن lemmatizing و stemming می توانیم ریشه کلمات را پیدا کنیم که تابع stemming زمان کمتری برای یافتن ریشه کلمات صرف می کند.



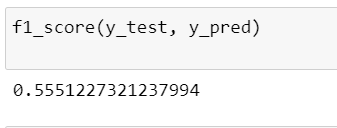
پس از تعریف کردن این توابع یک تابع به نام clean\_text داریم که همه توابع قبلی را به ترتیب روی متن ورودی اعمال می کند و نتیجه را بازمی گرداند :



در آخر هم با استفاده از توابع CountVectorizer و fit\_transform که برای آماده سازی دیتای text در ماشین لرنینگ استفاده می شوند ، پیش بینی را انجام می دهیم و معیار f1 را محاسبه می کنیم . دقت برای داده های train و test برابر 93 درصد می شود :



معیار f1 به دست آمده برای داده های test برابر 0.55 است :



معیار f1 به دست آمده برای داده های train برابر 0.56 است .

* **سوال سوم :**

برای حل این مسئله ابتدا به پیش پردازش داده ها می پردازیم.

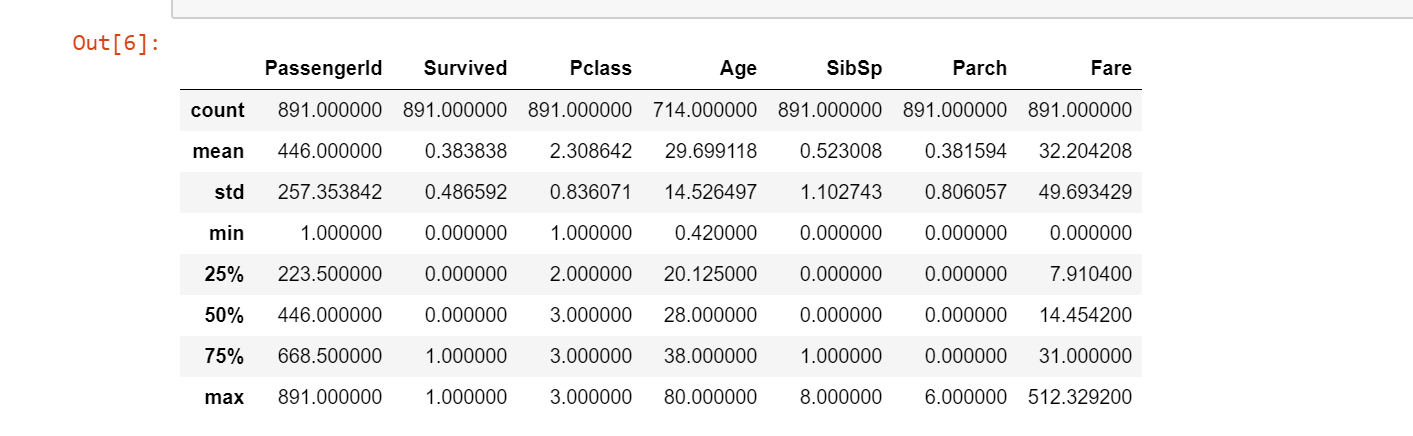
* Data extraction

در این قسمت ابتدا داده ها را لود کرده و یک نگاه کلی به داده ها داریم. مثلا یکی از کارهایی که در این مرحله می توان انجام داد توجه به فیچرهای موجود در دیتاست و درک مفهوم هر فیچر و مقادیر آن است. می توانیم پس از لود کردن دیتاست آموزشی با دستور data.head() پنج سطر اول دیتا را مشاهده کنیم :



* Data cleaning

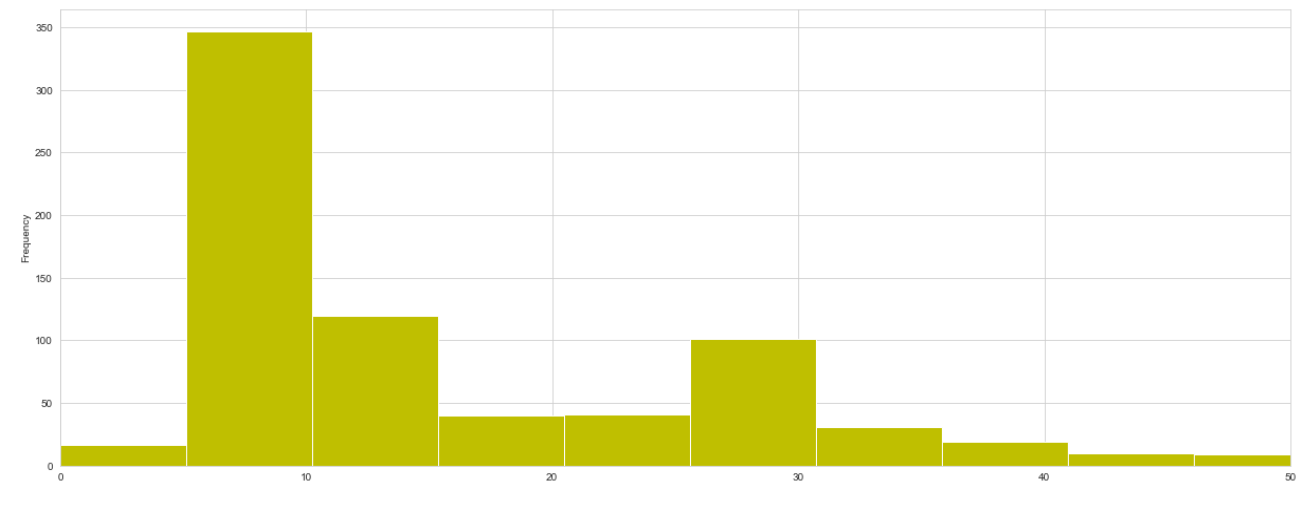
در این مرحله مقادیری از دیتاست که خالی هستند را پرمی کنیم. برای اینکه بفهمیم در کجای دیتاست و چه تعداد خانه ی خالی داریم از دستور data.describe() استفاده می کنیم. نتیجه این دستور روی دیتای train به شکل زیر است :



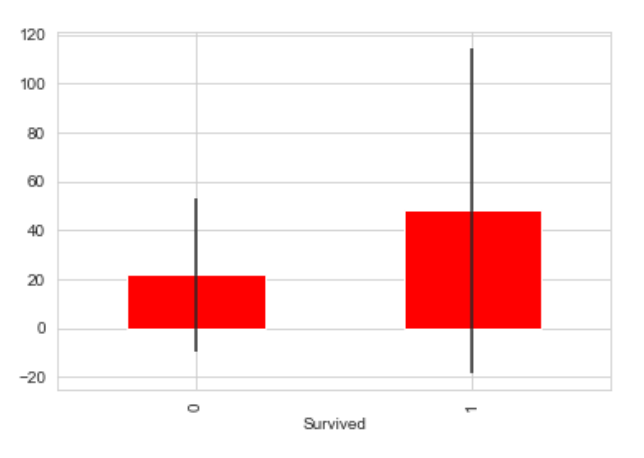
همان طور که در ردیف مربوط به count مشاهده می شود 891 مسافر داریم اما 714 سن ثبت شده است بنابراین 177 مقدار در ستون age خالی هستند. برای پرکردن این مقادیر گم شده ، می توانیم میانگین بقیه سن ها را قرار بدهیم اما این روش نسبت به outlier ها مقاوم نیست چون دیتاهای خیلی بزرگ یا خیلی کوچک می توانند باعث تغییر چشم گیری در میانگین شوند .  
برای ستون embarked نیز مقادیر خالی را با S که بیشترین تعداد تکرار را دارد پر می کنیم.  
برای ستون Fare مقادیر خالی را با میانه پر می کنیم ( که البته فقط در دیتای تست مقادیر خالی وجود داشت) . هم چنین تایپ آن را از float به integer تغییر می دهیم.  
هم چنین در این مرحله می توانیم ستون هایی را که در آنالیز کردن دیتا تاثیری ندارند حذف کنیم. مثلا PassengerId و Name تاثیری در نجات یافتن یا نیافتن ندارند. ستون cabin را هم حذف می کنیم چون مقادیر NaN زیادی و نمی تواند تاثیر زیادی روی نتیجه داشته باشد.  
در مورد متغیر Pclass نیز کلاس 3 را حذف می کنیم چون کمترین میانگین تعداد مسافران نجات یافته را دارد.

* رسم نمودار

برای اینکه هم بستگی های غیرقابل مشاهده میان فیچرها را بفهمم از رسم نمودار کمک می گیریم. نمودار زیر فراوانی بلیط ها بر حسب قیمت آن ها را نشان می دهد. مشاهده می شود که اکثر مسافران بلیطی با قیمتی بین 5 تا 10 دلار داشته اند.



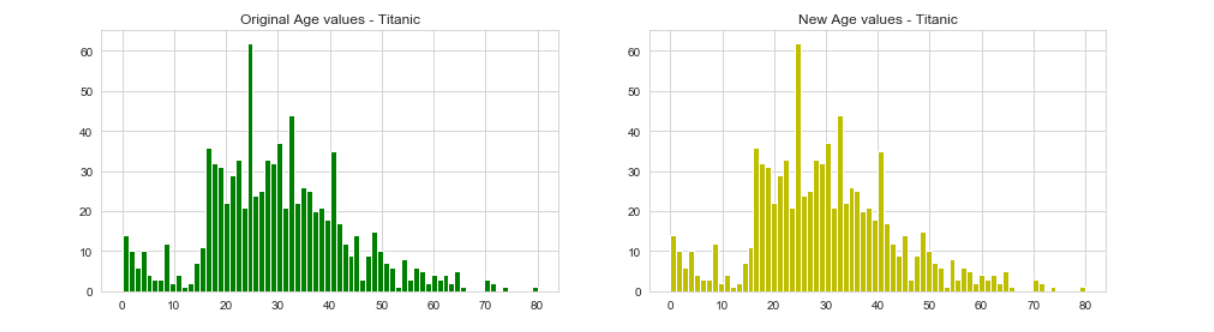
نمودار زیر ارتباط میان نجات یافتن و قیمت بلیط را نشان می دهد. کسانی که بلیط های گران قیمت تری خریده اند احتمال نجاتشان بیشتر است !



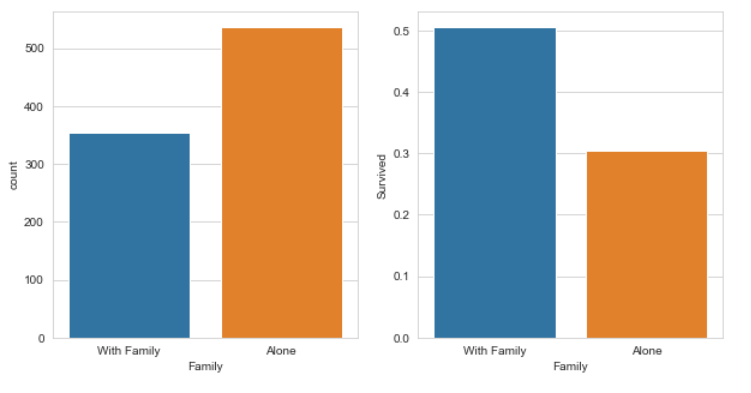
برای پیش پردازش ستون age ابتدا برای دیتاهای تست و train مقادیر میانگین ، انحراف معیار و تعداد NaN ها را تعیین می کنیم. سپس اعداد رندمی به تعداد مقادیر خالی سن در هر دو دیتا ، بین mean – std و mean + std تولید میکنیم و جاهای خالی را با این مقادیر پر می کنیم :

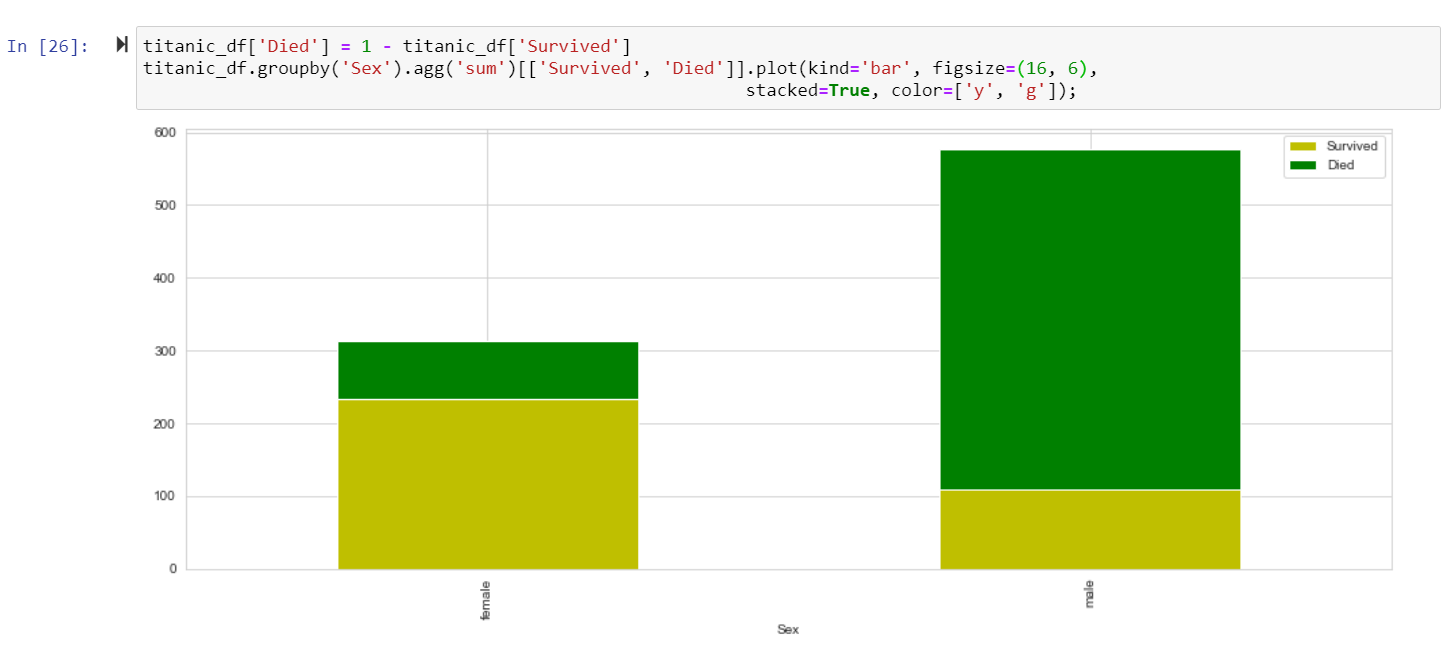


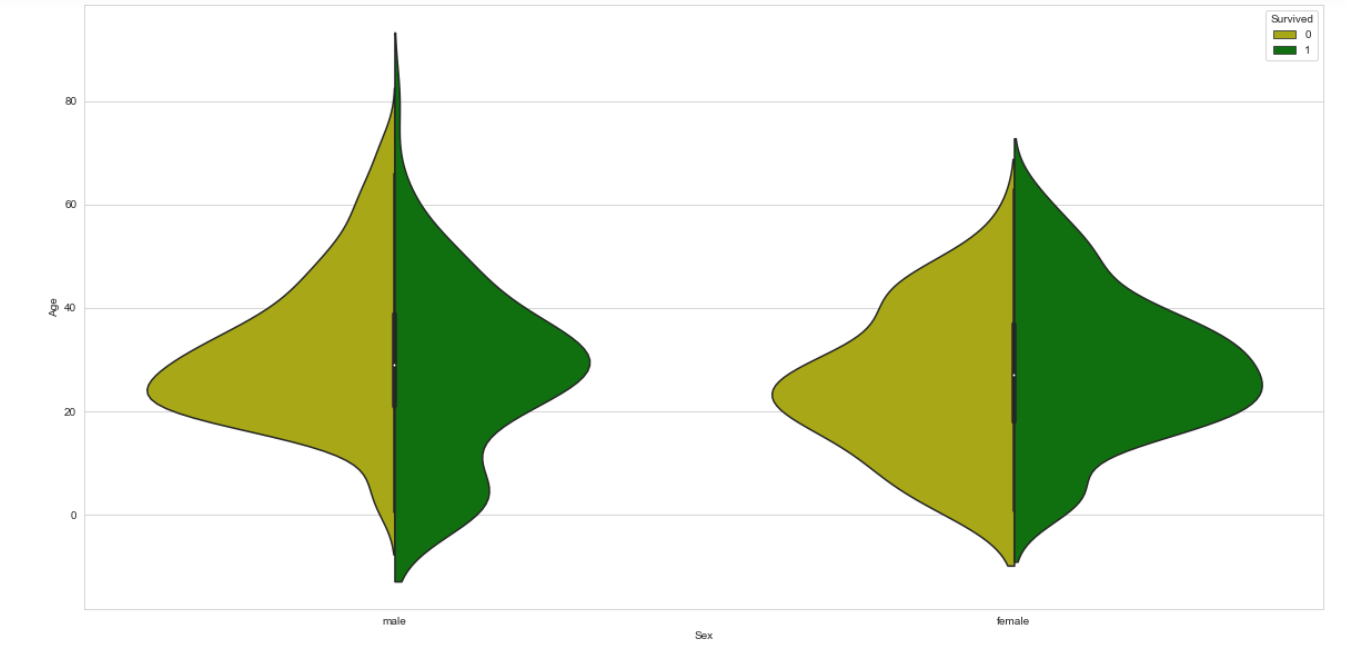
نمودار زیر مقادیر age را قبل و بعد از انجام این پیش پردازش نشان می دهد :



به جای دو ستون **Parch** و **SibSp** یک ستون به نام family قرار می دهیم و می خواهیم به طور کلی بررسی کنیم که آیا همراه بودن با خانواده در کشتی ، تاثیری در نجات یافتن دارد یا نه . نمودار سمت چپ در شکل زیر فراوانی افراد بر حسب همراه بودن با خانواده شان را نشان می دهد . می بینم که تعداد کسانی که تنها بوده اند بیشتر است. نمودار سمت راست ارتباط میان نجات یافتن و همراه بودن خانواده را نشان می دهد .مشاهده می شود که از جمعیت نجات یافتگان تعداد کسانی که با خانواده بیوده اند بیشتر از کسانی است که تنها بوده اند .

  
  
نمودار زیر ارتباط میان جنسیت و نجات یافتن را نشان می دهد. همان طور که مشاهده می شود درصد زنان نجات یافته بیشتر از زنانی است که جان خود را از دست داده اند اما در مورد مردان برعکس است.



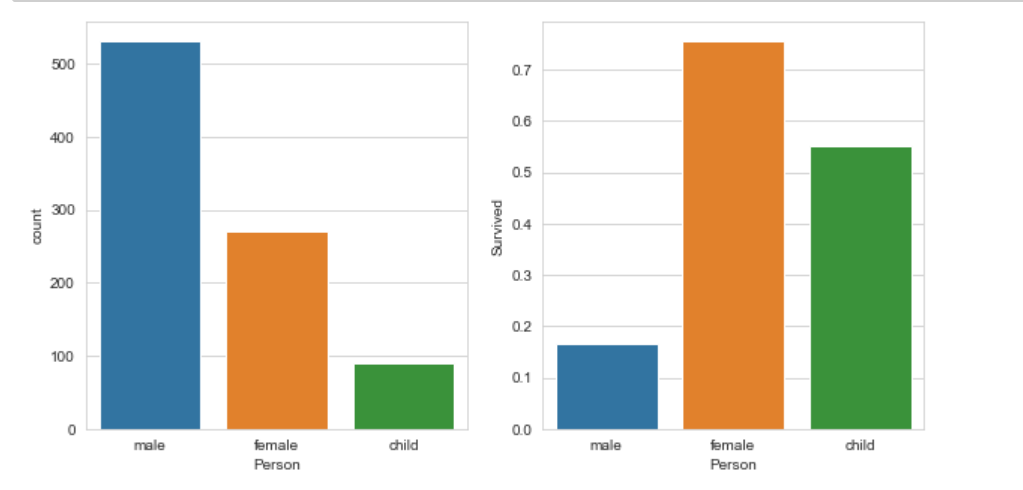


مشاهده می شود که تعداد مردان نجات یافته کمتر از زنان نجات یافته است ( باتوجه به اینکه قسمت سبز رنگ نمودار سمت راست بزرگتر از سمت چپ است). همچنین می توان مشاهده کرد که تعداد پسربچه های نجات یافته از آن هایی که نجات نیافته اند بیشتر است اما اکثر مردانی که سن بین 20 تا 40 سال داشته اند جان خود را از دست داده اند.

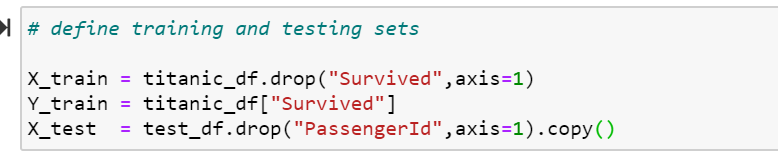
در مورد زنان نمی توان اظهار نظر دقیقی در مورد تاثیر سن داشت.

از آن جایی که بچه ها (کسانی که سن زیر 16 سال دارند) شانس بیشتری برای نجات داشته اند مسافران را به سه دسته مردان ، زنان و کودکان تقسیم بندی می کنیم:

  
  
نمودار سمت چپ در تصویر زیر نشان دهنده فراوانی افراد برحسب این سه کلاس است و نمودار سمت راست فراوانی افراد نجات یافته را بر اساس این سه دسته بندی نشان می دهد. بیشتر افرادی که نجات یافته اند از دسته ی زنان و کودکان بوده اند:



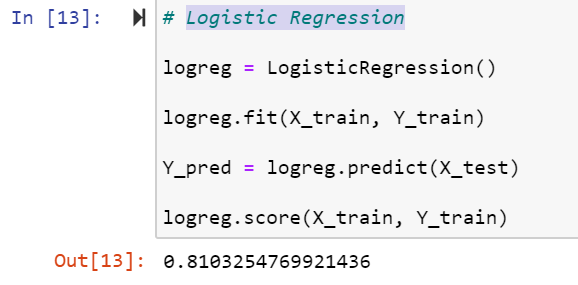
بعد از انجام این پیش پردازش ها حالا مجموعه هایtrain و test را تعریف می کنیم :



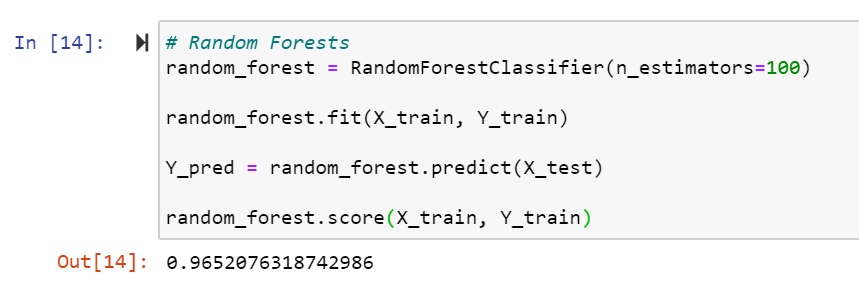
حالا با استفاده از الگوریتم های مختلف مدل می سازیم و دقت مدل ها را محاسبه می کنیم:

1. Logistic Regression

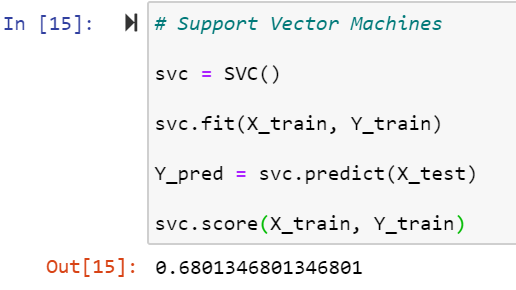
دقت : 0.810



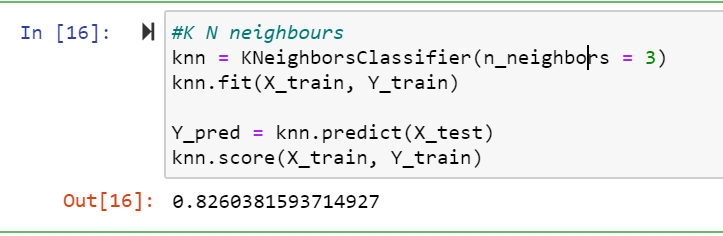
1. Random Forests  
   دقت : 0.965



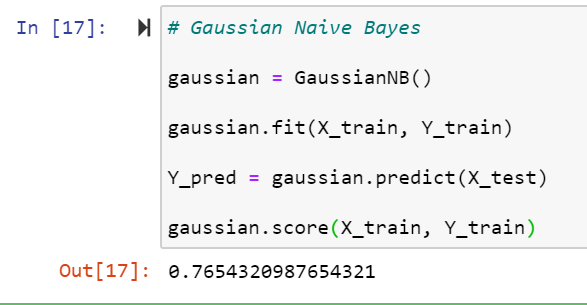
1. Support Vector Machines  
   دقت : 0.680



1. K N neighbours  
   دقت : 0.826



1. Gaussian Naive Bayes  
   دقت : 0.765



همان طور که مشاهده می شود Logistic Regression و Random Forests دارای بالاترین دقت هستند .