زهرا احمدنژاد (400422011) کلاس داده کاوی مجازی

Mobile Price Classification

در این دیتاست ما داده های فروش تلفن همراه شرکت های مختلف را داریم. میخواهیم رابطه بین ویژگی های تلفن همراه و قیمت فروش آنرا پیدا کنیم و مجبور نیستیم قیمت واقعی را پیدا کنیم، میتوانیم محدوده قیمت را به دست بیاوریم.

جواب سوال1)

سوال اول از ما خواسته شده تا روش انتخاب ویژگی forward selection را پیاده سازی کنیم.

انتخاب پیشرو (forward selection)

در روش forward selection یا انتخاب رو به جلو، ما با یک مدل تهی شروع می کنیم و مدل n را با استفاده از هر ویژگی (n) به صورت جداگانه آموزش میدهیم و شروع به تطبیق مدل با هرfeature جداگانه می کنیم وعملکرد را بررسی میکنیم.

متغیری که بهترین عملکرد را دارد و با حداقل p-value انتخاب می کنیم. بعد با آزمایش ترکیبی از ویژگی انتخاب شده قبلی با سایر ویژگی های باقی مانده، مدلی را با دو ویژگی متناسب میکنیم. فرآیند را تکرار میکنیم ودرهرمرحله یک فیچر یا ویژگی به فیچر قبلی انتخاب شده، اضافه میکنیم.

متغیری که بیشترین بهبود را ایجاد می کند انتخاب و نگه داشته می شود.

کل فرآیند را تا وقتی که بهبود قابل توجهی درعملکرد مدل ایجاد نشود تکرار میکنیم.

برای اینکه مدل موردنظرمان را بسازیم ابتدا باید x,y را تعریف کنیم.

# train

x = mpc\_train\_data.iloc[:,0:20]

y = mpc\_train\_data.iloc[:,-1]

ما اینجا x,y را روی داده train تعریف میکنیم اما چون به x,y تست هم نیاز دارم باید داده train را به دو بخش train , test تقسیم کنم.

from sklearn.model\_selection import train\_test\_split

#spliting the dataset into training and test set

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=1/4)

با این کد قسمت train و test داده را از هم جدا میکنم.

#Normalizing the data

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = pd.DataFrame(sc.fit\_transform(x\_train),columns = x\_train.columns)

x\_test = pd.DataFrame(sc.transform(x\_test),columns = x\_test.columns)

در این قسمت نرمال سازی دیتا را انجام میدم.

from sklearn import metrics

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

select\_feature = []

all\_feature = set(x.columns)

for k in range(5):

  kth\_auc = 0

  kth\_feature = None

  for f in (all\_feature - set(select\_feature)):

    tmp\_feature = deepcopy(select\_feature)

    tmp\_feature.append(f)

    model = LogisticRegression().fit(x\_train[tmp\_feature], y\_train)

    y\_pred = model.predict(x\_test[tmp\_feature])

    auc = metrics.accuracy\_score(y\_test, y\_pred)

    if kth\_auc < auc:

      kth\_auc = auc

      kth\_feature = f

  select\_feature.append(f)

در اینجا مدل را میسازم. از مدل logistic Regression استفاده میکنم. میخواهم 5 ویژگی انتخاب کنم. و در آخر کد هم از if استفاده کردم که هر مرحله بهترین f را انتخاب کرده و در kth\_feature قرار دهد.

و all feature = x.columns قرار دادم چون قیمت را هم داشت اما x قیمت ندارد.

print(select\_feature)

این کد فیچرهایی که در کد قبل انتخاب شده را به ما نشان میدهد.

جواب سوال2)

قسمت آخر سوال دوم از ما خواسته شده تا f1\_score , recall , precision را به دست آوریم.

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,y\_pred, zero\_division=1))

با استفاده از این کد مقادیر خواسته شده را به دست می آوریم.

recall را به عنوان تعداد مثبت های واقعی (true positive) تقسیم بر تعداد مثبت های واقعی به اضافه تعداد منفی های کاذب (true positives plus false negatives) تعریف می کنیم.

precision توانایی یک مدل طبقه بندی برای شناسایی تنها نقاط داده مربوطه است. از نظر ریاضی، precision تعداد مثبت های واقعی (true positives) تقسیم بر تعداد مثبت های واقعی به اضافه تعداد مثبت های کاذب (true positives plus false positives) تعریف میشود.

در مواردی که می‌خواهیم یک ترکیب بهینه از preciosion and recall پیدا کنیم، می‌توانیم این دو معیار را با استفاده از score F1 ترکیب کنیم.

جواب3)

یکی از روش‌هایی که برای کاهش حجم و پیچیدگی داده‌ها استفاده می‌شود، روش Principal Component Analysis یا PCA است. در آنالیز PCA، هدف ما یافتن الگوهای مختلف در داده‌های مورد نظر است؛ در این آنالیز سعی بر آن است که همبستگی (correlation) میان داده‌ها به دست آید. اگر در دو یا چند بعد مشخص، بین داده‌های ما، همبستگی شدیدی وجود داشته باشد، می‌توان آن بعدها را به یک بعد تبدیل نمود؛ به این ترتیب و با کاهش بعد، پیچیدگی و حجم داده‌ها به شدت کاهش می‌یابد. بنابراین هدف پیدا کردن جهت (بعد) با بیش‌ترین واریانس داده‌ها و کاهش بعد است؛ به صورتی که کم‌ترین میزان داده‌ی بااهمیت از دست رود.

یک مزیت جانبی مهم این تکنیک این است که نمرات PCA حاصله همبستگی ندارند. این ویژگی برای تکنیک‌های مدل‌سازی که نیاز دارند ویژگی‌ها نسبتاً همبسته نباشند، بسیار مفید است.

زمانی که داده های موجود از یک یا چند دسته از ویژگی ها که حاوی اطلاعات اضافی هستند (به عنوان مثال، ویژگی هایی که با یکدیگر همبستگی زیادی دارند) تشکیل شده باشد، PCA ابزار مفیدی است.

رگرسیون مؤلفه اصلی، که فرآیند ابتدا کاهش بعد و سپس پسرفت امتیازات مؤلفه جدید در پاسخ را توصیف می کند، یک فرآیند گام به گام است.

بنابراین با استفاده از این کد، pca را در ابتدا import کرده و سپس روی x train , x test تعریف میکنیم.

From sklearn.decomposition import PCA

pca = PCA(n\_components=len(select\_feature))

pca\_x\_train = pca.fit\_transform(x\_train)

pca\_x\_test = pca.transform(x\_test)

و بعد از آن model , y\_pred را تعریف میکنیم.

model = LogisticRegression().fit(pca\_x\_train, y\_train)

y\_pred = model.predict(pca\_x\_test)

جواب4)

f1 score , recall , precision را برای دیتاست تغییر یافته محاسبه میکنیم.

print(classification\_report(y\_test,y\_pred, zero\_division=1))

جواب6\_آ) مهندسی ویژگی

برای binning ابتدا از یک نمودار استفاده کرده تا ببینیم توزیع چگونه است.

x\_train.battery\_power.hist()

من چهار اندازه مختلف برای بین ها در نظر گرفتم.

min\_value = x\_train['battery\_power'].min()

max\_value = x\_train['battery\_power'].max()

print('min:', min\_value, ', max:', max\_value)

bins = np.linspace(min\_value,max\_value,4)

print(bins)

در اینجا مقدار min , max را به ما نشان میدهد و همینطور 4 بین موردنظر خواسته شده را.

x\_train['battery\_power'] = pd.cut(x\_train['battery\_power'], bins=bins)

x\_train['battery\_power'].value\_counts()

این کد هم تعداد هرکدام از دسته ها را به ما نشان میدهد.

جواب سوال8)

بوت استرپینگ یک روش آماری است که یک مجموعه داده واحد را مجدداً نمونه‌برداری می‌کند تا نمونه‌های شبیه‌سازی شده زیادی ایجاد کند. این فرآیند امکان محاسبه خطاهای استاندارد، فواصل اطمینان و آزمون فرضیه ها را فراهم می کند.

bootstrapping , cross validation روش‌های نمونه‌گیری مجدد هستند. بوت استرپ مجدداً با جایگزینی نمونه‌برداری می‌کند (و معمولاً مجموعه‌های داده «جایگزین» جدیدی با تعداد موارد مشابه مجموعه داده اصلی تولید می‌کند). به دلیل ترسیم با جایگزینی، یک مجموعه داده بوت استرپ ممکن است حاوی چندین نمونه از همان موارد اصلی باشد و ممکن است دیگر موارد اصلی را کاملا حذف کند.

Cross validation جایگزینی مجدد نمونه می شود و بنابراین مجموعه داده های جایگزینی تولید می کند که کوچکتر از مجموعه داده اصلی هستند. این مجموعه داده ها به روشی سیستماتیک تولید می شوند به طوری که پس از یک عدد از پیش تعیین شده k از مجموعه داده های جایگزین، هر یک از n مورد اصلی دقیقاً یک بار کنار گذاشته شده است.

هدف اصلی cross validation اندازه گیری عملکرد یک مدل است. در مقابل، bootstrapping در درجه اول برای ایجاد توابع توزیع تجربی برای طیف گسترده ای از آمارها استفاده می شود.

Cross validation مجموعه داده موجود را برای ایجاد مجموعه داده های متعدد تقسیم می کند و روش بوت استرپینگ از مجموعه داده اصلی برای ایجاد مجموعه داده های متعدد پس از نمونه برداری مجدد با جایگزینی استفاده می کند. هنگامی که برای اعتبارسنجی مدل استفاده می شود، بوت استرپینگ به اندازه validation Cross قوی نیست. Bootstrapping بیشتر در مورد ساخت مدل های گروه یا فقط تخمین پارامترها است.

جواب سوال9)

آزمون‌های آماری تقریبی برای مقایسه الگوریتم‌های یادگیری طبقه‌بندی نظارت شده، انجام 5×2cv را پیشنهاد می‌کند، که عبارت است از تکرار 5 بار در 2 برابر و سپس محاسبه میانگین.