

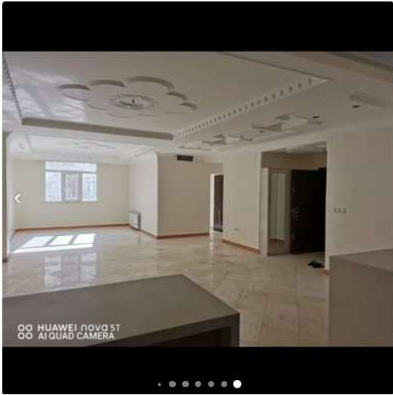
فهرست

2.....	مرحله صفر - شرح دیتاست
3.....	مرحله اول - شناسایی و پاکسازی داده
4.....	فیلد داده‌های اساسی و مهم را شناسایی کنید
4.....	داده‌ها را جمع آوری کنید
4.....	از مقادیر تکراری صرف نظر کنید
4.....	مقادیر خالی را برطرف کنید
4.....	فرآیند پاک سازی را استاندارد کنید
4.....	داده‌ها را مرور کنید و تطبیق دهید
4.....	مرور دیتاست قیمت خانه های تهران
5.....	داده پرت یا outlier
7.....	راهنمای Panda
8.....	راهنمای Regex
9.....	Dependent and independent variables
9.....	Highly correlated features
9.....	Local Outlier Factor Calculation
10.....	One Class SVM
11.....	مرحله دوم - توزیع نرمال, Skewness
13.....	power transformation
14.....	StandardScaler
15.....	مرحله سوم - کدگذاری دیتا
16.....	Dummy Variable Trap in Regression Models
17.....	Multi-collinearity
17.....	مرحله چهارم - تقسیم داده
17.....	مرحله پنجم - الگوریتمها و شرح و مقایسه با پارامتر
17.....	● رگرسیون خطی
18.....	○ مزایای رگرسیون خطی
18.....	○ معایب رگرسیون خطی

19.....	●درخت تصمیم
19.....	○اجزای اصلی درخت تصمیم
19.....	○معیارهای انتخاب مشخصه برای انشعاب درخت تصمیم
20.....	●جنگل تصادفی (Random Forest)
20.....	درخت تصمیم، بلوک سازنده جنگل تصادفی
22.....	●نزدیک‌ترین همسایگی (k-Nearest Neighbors)
23.....	k-fold Cross-Validation چیست
24.....	GridSearchCV
25.....	مرحله هفتم -مقایسه نتایج و تفسیر نتایج
26.....	● Confusion Matrix
28.....	● F1 score
29.....	● Average_precision_score
29.....	Area Under the Curve
30.....	adjusted_rand_score
30.....	● R2
31.....	● MAE Mean Absolute Error
31.....	● MAPE-Mean Absolute Percentage Error
32.....	● Root Mean Square Error -RSME
33.....	Bias–variance tradeoff
33.....	● Bias Error
33.....	● Variance Error
34.....	● Overfitting vs. Underfitting
36.....	مرحله هشتم -افزایش نمونه ها در صورت نیاز
38.....	سخن پایانی
38.....	مراجع

مرحله صفر - شرح دیتاست

دیتاست مورد نظر شامل رکوردهایی در خصوص قیمت فروش املاک در تهران میباشد. یک ملک شامل منطقه - مساحت - قیمت و امکانات پارکینگ و آسانسور است همینطور ستون قیمت به دلار نیز وجود دارد. باید توجه داشت که پارکینگ در بسیاری نقاط به صورت جداگانه محاسبه میشود.



۱۱۴ متری، تک واحدی، دو خوابه، کوچه اکبر رصاف

۱ ساعت پیش در تهران، پرستار | فروش آپارتمان

چت اطلاعات تماس

متراژ	۱۱۴
ساخت	۱۳۹۱
اتاق	۲

قیمت کل	۳,۱۰۰,۰۰۰,۰۰۰ تومان
قیمت هر متر	۳۵,۹۶۵,۰۰۰ تومان
طبقه	۳

ویژگی‌ها و امکانات

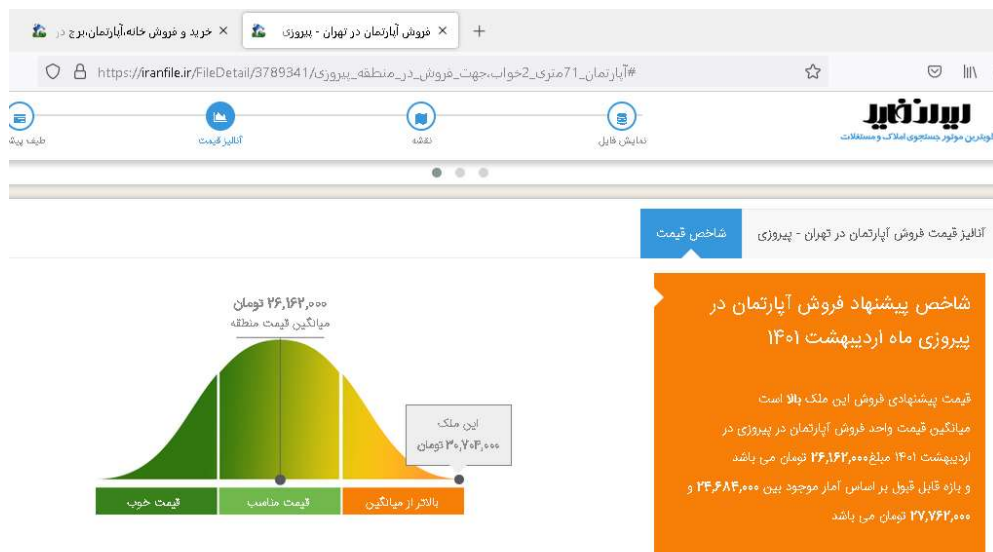
آسانسور | پارکینگ | انباری ندارد

توضیحات

پاداشقت شما...

شکل ۱- یک فایل نمونه برای واحد ملکی

هدف از این تمرین تهیه یک مدل با قابلیت پیش بینی قیمت با توجه به ادرس و مشخصات بتوان ملک را ارزیابی کرد. مانند سایت iranfile که در زیر آورده شده است



شکل ۲- آنالیز و پیشگویی قیمت یک ملک

مرحله اول – شناسایی و پاکسازی داده

به فرآیند رفع یا حذف داده های نادرست، خراب، با قالب بندی اشتباه، کپی یا ناقص در یک مجموعه داده، پاکسازی داده می گویند. برای انجام این فرایند مراحل زیر را دنبال میکنیم

فیلد داده‌های اساسی و مهم را شناسایی کنید

- ✓ هر چند این روزها کسب و کارها به داده‌های بیشتری دسترسی دارند، اما همه آنها به یک اندازه مفید نیستند. اولین گام در پاکسازی داده‌ها این است که تعیین کنید کدام نوع داده یا فیلد داده برای یک پروژه یا فرآیند خاص ضروری است.

داده‌ها را جمع آوری کنید

- ✓ بعد از شناسایی فیلد داده‌های مدنظر، می‌توانید داده‌های موجود در آنها را جمع آوری و سازماندهی کنید.

از مقادیر تکراری صرف نظر کنید

- ✓ بعد از جمع آوری داده‌ها، زمان رفع اشتباهات و خطاهاست. حالا زمان آن رسیده که مقادیر تکراری را شناسایی و حذف کنید.

مقادیر خالی را برطرف کنید

- ✓ ابزار پاکسازی داده‌ها، هر فیلد را برای مقادیر از دست رفته، جستجو کرده و سپس آن مقادیر را برای ایجاد یک مجموعه کامل داده و جلوگیری از شکاف در اطلاعات پر می‌کند.

فرآیند پاک سازی را استاندارد کنید

- ✓ کدام داده بیشتر مورد استفاده قرار می‌گیرد؟
- ✓ چه زمانی به آن داده نیاز دارید؟
- ✓ چه کسی مسئول حفظ فرآیند است؟
- ✓ هر چند وقت یکبار باید داده‌های خود را پاکسازی کنید؟ (روزانه - هفتگی - ماهانه)

داده‌ها را مرور کنید و تطبیق دهید

- ✓ لازم است برای بررسی فرآیند پاکسازی داده‌ها، بازه‌های زمانی منظم (مثلاً هر هفته یا هر ماه) در نظر بگیرید. به این ترتیب می‌توانید ملاحظه کنید کدام قسمت به خوبی کار می‌کند؟ کجا نیاز به تغییر و پیشرفت دارد؟ ایرادات آشکار فرآیند کجاست؟

مرور دیتاست قیمت خانه های تهران

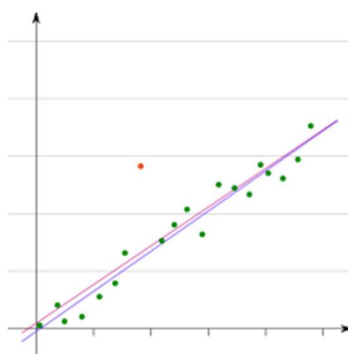
df1 - DataFrame								
Index	Area	Room	Parking	Varehouse	Elevator	Address	Price	Price(USD)
0	63	1	True	True	True	Shahran	185000000.000000	61666.670000
1	60	1	True	True	True	Shahran	185000000.000000	61666.670000
2	79	2	True	True	True	Pardis	55000000.000000	18333.330000
3	95	2	True	True	True	Shahrake Qods	90250000.000000	30083.330000
4	123	2	True	True	True	Shahrake Gharb	700000000.000000	233333.330000
5	70	2	True	True	False	North Program Organization	205000000.000000	68333.330000
6	87	2	True	True	True	Pardis	60000000.000000	20000.000000
7	59	1	True	True	True	Shahran	215000000.000000	71666.670000
8	54	2	True	True	False	Andisheh	493000000.000000	16433.330000
9	71	1	True	True	True	West Ferdows Boulevard	237000000.000000	79000.000000
10	68	2	True	True	True	West Ferdows Boulevard	245000000.000000	81666.670000
11	64	1	True	True	True	Narmak	210000000.000000	70000.000000
12	54	1	False	True	True	Narmak	169000000.000000	56333.330000
13	136	3	True	True	True	Saadat Abad	1100000000.000000	366666.670000
14	95	2	True	True	True	Zafar	500000000.000000	166666.670000
15	63	1	False	True	False	Islamshahr	570000000.000000	19000.000000
16	155	3	True	True	True	Narmak	670000000.000000	223333.330000
17	64	2	False	True	False	Pirouzi	145000000.000000	48333.330000
18	140	3	True	True	True	West Ferdows Boulevard	640000000.000000	213333.330000

شکل 3- نمایش دیتافریم برای قیمت خانه

برای لود دیتا به شکل زیر عمل میشود

```
df1 = pd.read_csv('housePrice.csv')
```

داده پرت یا outlier



شکل 4- مثال داده پرت

داده پرت داده‌ای است که تفاوت قابل ملاحظه‌ای با بقیه اعضای نمونه‌ای که در آن اتفاق افتاده است داشته باشد. داده‌ای پرت (قرمز) که ظاهراً با دیگر داده‌ها نمی‌خواند، رگرسیون خطی آبی بدون در نظر گرفتن تأثیر آن ترسیم شده و خط قرمز با به حساب آوردن آن.

در کد روشهای مختلفی برای حذف دیتای پرت استفاده شده است. مهمترین آن استفاده از یک دیتاست کمکی است. دیتاست کمکی این امکان را ایجاد میکند تا به طور موثری دیتاهای پرت را پیدا کرده و حذف کنیم.

Index	Address	Loc
0	Shahrar	5
1	Pardis	8
2	Shahrak	2
3	North	8
4	Andisheh	7
5	West	8
6	Narmak	4
7	Saadat	2
8	Zafar	3
9	Islanshehr	8
10	Pirouzi	13
11	Moniriyeh	11
12	Velenjak	1
13	Amirieh	11
14	Southern	8
15	Katoukhil	18

شکل 5- خلاصه نمایش دیتاست مناطق تهران

دیتاست مناطق تهران این امکان را ایجاد میکند که آدرسهای خارج از محدوده را حذف کنیم. مثلاً اگر اسلامشهر وجود دارد خارج از تهران بوده آنرا حذف میکنیم.

```
df2 = pd.read_csv('Tehran_Aria.csv')
house_price_raw = pd.merge(df1, df2, how='inner', on = 'Address')
house_price_raw = house_price_raw[house_price_raw['Loc'] != 0]
```

همینطور این امکان برای ما ایجاد میشود که فقط یک منطقه را بررسی کنیم. میتوان اطلاعات مربوط به دیتاست را به طریق زیر نمایش داد

```
house_price_raw.info()
```

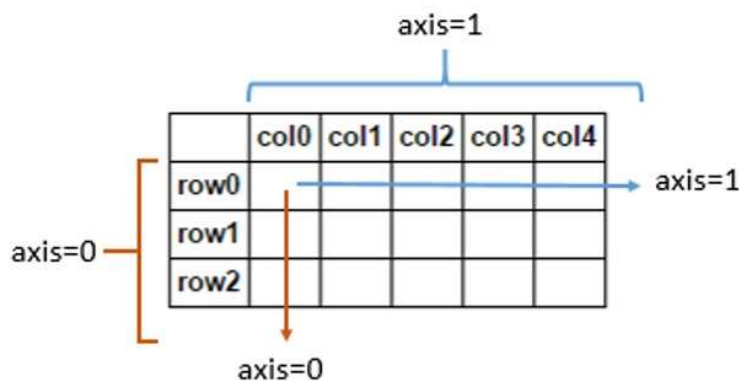
Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Area	1679 non-null	object
1	Room	1679 non-null	int64
2	Parking	1679 non-null	bool
3	Warehouse	1679 non-null	bool
4	Elevator	1679 non-null	bool
5	Address	1679 non-null	object
6	Price	1679 non-null	float64
7	Price(USD)	1679 non-null	float64
8	Loc	1679 non-null	int64

راهنمای Panda

از کتابخانه‌های متن‌بازی است که برای کار با داده‌هایی با ساختار رابطه‌ای (rational) یا برچسب‌گذاری شده ایجاد شده است. این کتابخانه ساختار داده‌های متنوعی به همراه امکان اعمال عملیات عددی روی این داده‌ها را فراهم می‌کند و به خوبی می‌تواند با سری‌های زمانی کار کند. Pandas بر مبنای کتابخانه‌ی NumPy ساخته شده است و بسیاری از ساختارهای NumPy در این کتابخانه استفاده شده و گسترش یافته‌اند. در ادامه به آموزش کتابخانه Pandas می‌پردازیم. مزایای این کتابخانه شامل موارد زیر است:

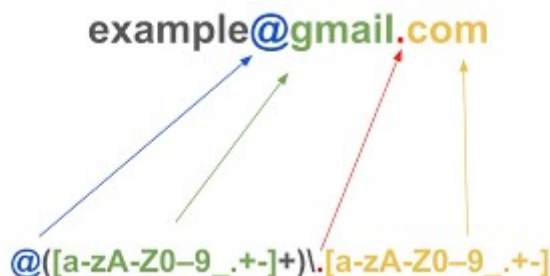
- سرعت و کارایی بالا در کار با داده‌ها.
- امکان بارگذاری داده‌ها از فایل‌های متفاوت.
- کنترل راحت ویژگی‌هایی که مقداردهی نشده‌اند. به عبارتی این مقادیر با NaN مقداردهی شده‌اند
- قابلیت تغییر اندازه: ستون‌هایی می‌توانند به داده‌ها اضافه شوند و یا از آن‌ها حذف شوند.
- ادغام (merge) و اتصال (join) مجموعه داده‌ها.
- تغییر شکل داده به طور منعطف.
- فراهم کردن امکان کار با سری‌های زمانی.
- امکان گروه‌بندی داده‌ها با توجه به اهداف کاربردی.



شکل 6- ساختار پاندا

راهنمای Regex

عبارت‌های منظم (Regular Expressions) که اصطلاحاً regex نامیده می‌شوند در زمان استخراج اطلاعات از هر متنی کاملاً مفید هستند. این عبارت‌ها برای جستجو و یافتن مطابقت یک یا چند الگوی جستجوی خاص مورد استفاده قرار می‌گیرند. بدین ترتیب می‌توان توالی خالصی از کاراکترهای ASCII یا یونیکد را یافت. زمینه‌های کاربرد regex از اعتبارسنجی تا تجزیه/جایگزینی رشته‌ها، ترجمه داده‌ها به قالب‌های دیگر و وب اسکرپینگ متفاوت است.



شکل 7- مثال برای ریجکس

در برنامه از ریجکس برای پاکسازی دیتا استفاده شده است. خطوط زیر فضای خالی برای فیلد "مساحت" ملک را حذف نموده و سپس تمامی ارقام دو رقمی به بالا را از این فیلد استخراج میکند

```
dataset['Area'] = dataset['Area'].str.strip()
dataset['Area'] = dataset['Area'].astype(str).str.extract('([1-9][0-9]+)', expand=False)
```

برای تعداد اتاق هم یک تا نه قابل قبول است

```
dataset['Room'] = dataset['Room'].astype(str).str.extract('([1-9])', expand=False)
```

همینطور بقیه خطوط هم از منطق مشابه ای استفاده میکنند

برای حذف مقادیر نال دستور زیر اجرا میشود

```
dataset.dropna(inplace = True)
dataset = dataset.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

و تنها پس از آن میتوان مقادیر را به شکل عددی درآورد وگرنه صفر فرض میشوند

```
dataset['Area'] = dataset['Area'] = pd.to_numeric(dataset['Area'], errors='coerce')
dataset['Price'] = pd.to_numeric(dataset['Price'])
dataset['Room'] = pd.to_numeric(dataset['Room'])
```

و همچنین مقادیر Bool را به صورت عدد تبدیل کرد

```
boolean_features = ['Parking', 'Warehouse', 'Elevator']
dataset[boolean_features] = dataset[boolean_features].astype('int64')
```


Dependent and independent variables

متغیر مستقل علت است. مقدار آن مستقل از سایر متغیرهای مطالعه شما است. متغیر وابسته اثر است. مقدار آن به تغییرات متغیر مستقل بستگی دارد مثلاً آسانسور داشتن ملک (متغیر مستقل) روی قیمت آن تاثیر دارد ولی با قیمت ملک نمیشود گفت آسانسور دارد یا نه این اصلاً معنی ندارد. همیشه مجموعه ای از متغیرهای غیروابسته وجود دارد که برای پیش گویی یک متغیر وابسته استفاده میشود.

Highly correlated features

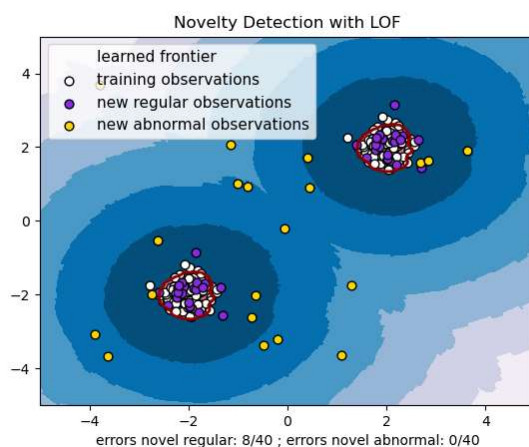
ویژگی های با همبستگی بالا بیشتر به صورت خطی وابسته هستند و از این رو تقریباً همان تأثیر را روی متغیر وابسته دارند. بنابراین، وقتی دو ویژگی دارای همبستگی بالایی هستند، می‌توانیم یکی از دو ویژگی را حذف کنیم – مانند فیلد قیمت و قیمت به دلار برای حذف ستون قیمت دلار به شکل زیر عمل میکنیم.

```
dataset = dataset.drop(columns = ['Price(USD)'])
```

Local Outlier Factor Calculation

Local Outlier Factor یک الگوریتم برای تشخیص ناهنجاری ها در داده های مشاهده است. اندازه گیری امتیاز چگالی محلی هر نمونه و وزن دادن به امتیازات آنها مفهوم اصلی الگوریتم است. با مقایسه امتیاز نمونه با همسایگانش، الگوریتم عناصر با چگالی کمتر را به عنوان ناهنجاری در داده ها تعریف می کند.

عامل دورافتاده محلی مبتنی بر مفهوم چگالی محلی است، جایی که محل با k نزدیکترین همسایه، که فاصله آنها برای تخمین چگالی استفاده می شود. با مقایسه چگالی محلی یک جسم با چگالی محلی همسایگانش، یک می تواند مناطقی با تراکم مشابه و نقاطی را که چگالی کمتری نسبت به همسایگان خود دارند شناسایی کند.



شکل 8- LOC

در برنامه خطوط زیر موجب حذف داده های پرت توسط LOC میشوند

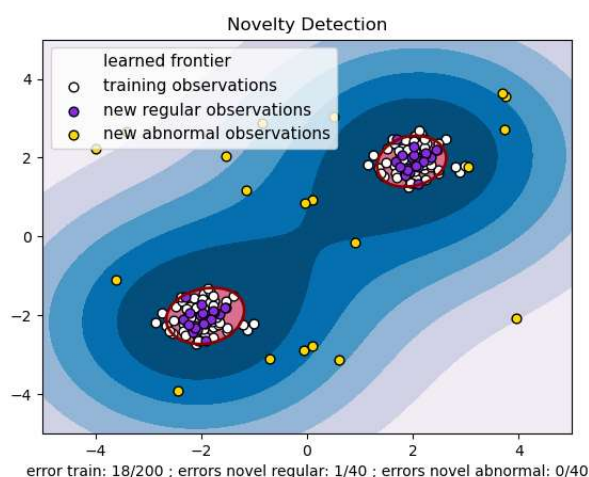
```
from sklearn.neighbors import LocalOutlierFactor
df = dataset[["Area", "Price"]]
model = LocalOutlierFactor(n_neighbors=20, contamination=.03).fit(df)
y_pred = model.fit_predict(df)
outlier_index = where(y_pred==-1)
outlier_values = df.iloc[outlier_index]

total_outliers = np.asarray(outlier_index )
total_outliers =total_outliers.flatten()
```

One Class SVM

ناهنجاری یا anomaly detection یا خارج از محدوده قرار گرفتن میتواند توسط SVM هم تشخیص داده شود. One-class SVM یک الگوریتم بدون نظارت است که یک تابع تصمیم گیری را برای تشخیص تازگی می آموزد: طبقه بندی داده های جدید به عنوان مشابه یا متفاوت با مجموعه آموزشی.

One Class SVM برای طبقه بندی، یک ابرصفحه حداکثر حاشیه را پیدا می کند که نمونه های مثبت را از نمونه های منفی جدا می کند. SVM یک کلاس یک ابر صفحه را پیدا می کند که مجموعه داده داده شده را از منبع جدا می کند به طوری که ابر صفحه تا حد امکان به نقاط داده نزدیک باشد. توجه داشته باشید که معمولاً از هسته RBF برای جا دادن یک مرز غیر خطی در اطراف ناحیه متراکم مجموعه داده استفاده می شود که نقاط باقیمانده را به عنوان نقاط پرت جدا می کند.



شکل 9-OneClassSVM

در برنامه میتوان از خطوط زیر هم برای حذف دیتای پرت با OneClassSVM استفاده کرد و باید این قسمت را uncomment کرد

```
# from sklearn.svm import OneClassSVM
# df = dataset[["Area", "Price"]]
# model = OneClassSVM(kernel = 'rbf').fit(df)
# y_pred = model.predict(df)
# outlier_index = where(y_pred == -1)
# # filter outlier values
# outlier_values = df.iloc[outlier_index]

# # visualize outputs
# plt.scatter(dataset["Area"], dataset["Price"])
# plt.scatter(outlier_values["Area"], outlier_values["Price"], c = "r")

# total_outliers = np.asarray(outlier_index )
# total_outliers =total_outliers.flatten()
```

یا اینکه از روال زیر برای حداکثر و حداقل استفاده کرد و نقاط پرت را شناسایی کرد.

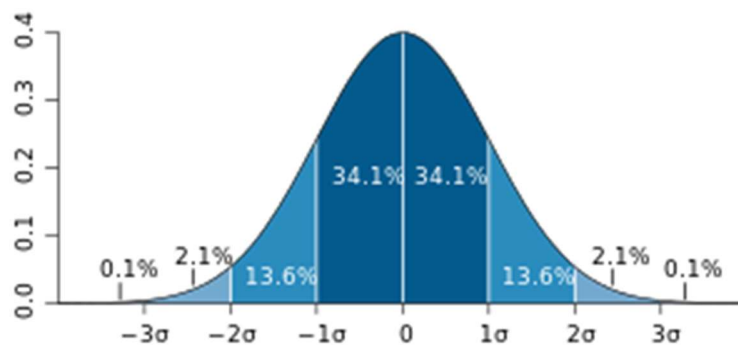
```
# def lower_upper(x):
#     Q1 = np.percentile(x, 25)
#     Q3 = np.percentile(x, 75)
#     IQR = Q3 - Q1
#     lower = Q1 - 1.5 * IQR
#     upper = Q3 + 1.5 * IQR

#     return lower, upper
# lower_area, upper_area = lower_upper(dataset['Area'])
# lower_price, upper_price = lower_upper(dataset['Price'])

# area_outliers = np.where(dataset['Area'] > upper_area)
# price_outliers = np.where(dataset['Price'] > upper_price)
# total_outliers = np.union1d(area_outliers, price_outliers)
```

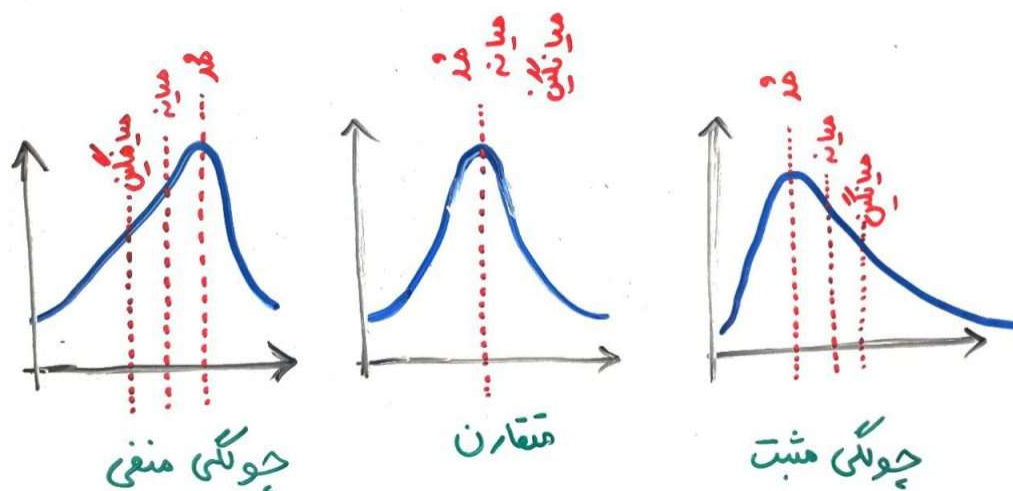
مرحله دوم - توزیع نرمال , Skewness

توزیع نرمال یک نوع با خصوصیات زیر است . تقریباً ۶۸٪ از کل اعدادی که از یک توزیع نرمال گرفته شوند، فاصله‌ای برابر یا کمتر از یک برابر انحراف معیار توزیع نسبت به میانگین توزیع دارند. تقریباً ۹۵٪ از کل اعدادی که از یک توزیع نرمال گرفته شوند، فاصله‌ای برابر یا کمتر از دو برابر انحراف معیار توزیع نسبت به میانگین توزیع دارند. تقریباً ۹۹/۷٪ کل اعداد نیز در فاصله کمتر از سه برابر انحراف معیار توزیع نسبت به میانگین توزیع قرار می گیرند.



شکل 10-توزیع نرمال یا گاوسی

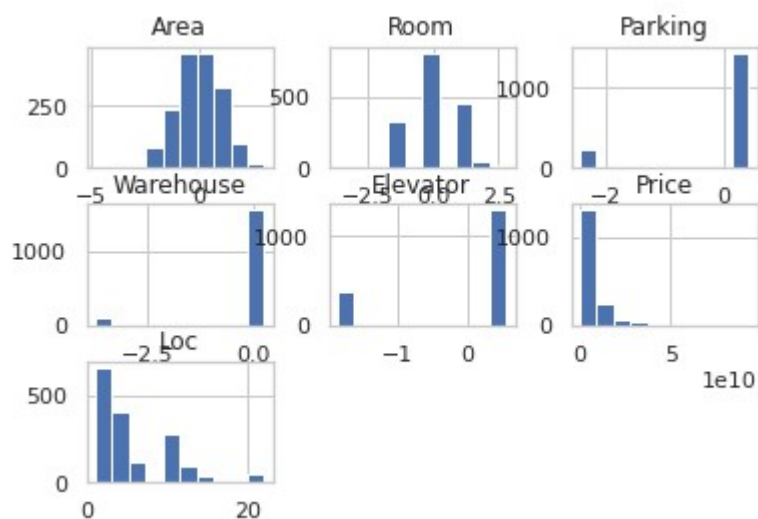
اگر توزیع داده‌ها به سمت راست یا چپ کشیده شده باشد، می‌گوییم داده‌ها چولگی یا همان skewness دارند.



شکل 11- چوگی داده ها

Lower Bound	Upper Bound	Result
$x < -1.0$	$x > +1.0$	highly skewed
$-0.5 < x < -1.0$	$0.5 < x < 1.0$	moderately skewed
$-0.5 < x < 0.5$	$-0.5 < x < 0.5$	approximately symmetric

شکل 12- تفسیر چوگی داده



شکل 13- توزیع داده ها در دیتاست

برای مشاهده Skewness دیتاست به شکل زیر عمل میکنیم.

```
dataset.skew()
```

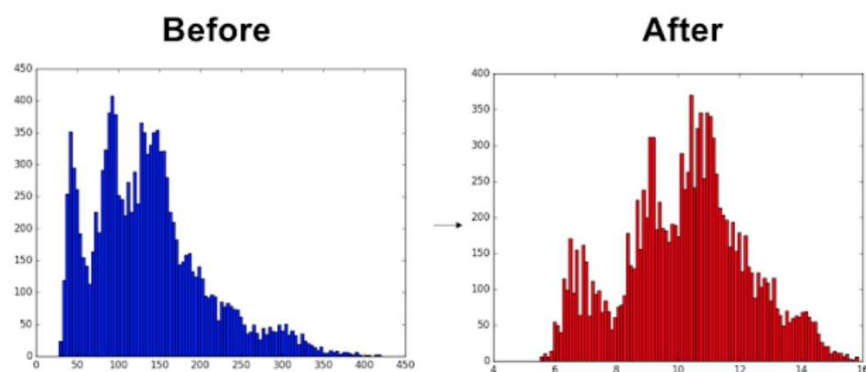
```
Area      3.823597
Room      0.529265
Parking   -2.020392
Warehouse -3.474946
Elevator  -1.304222
Price     3.813091
Loc       1.177199
```

power transformation

خانواده‌ای از تبدیل‌های پارامتریک و یکنواخت هستند که برای شبیه‌تر کردن گاوسی‌تر داده‌ها اعمال می‌شوند. این برای مدل‌سازی مسائل مربوط به ناهمسانی (واریانس غیر ثابت)، یا سایر موقعیت‌هایی که نرمال بودن مورد نظر است، مفید است. بسیاری از الگوریتم‌های یادگیری ماشین زمانی که توزیع متغیرها گاوسی باشد عملکرد بهتری دارند. به یاد داشته باشید که مشاهدات هر متغیر ممکن است از یک توزیع احتمال گرفته شده باشد. گاوسی یک توزیع رایج با شکل زنگ آشنا است. آنقدر رایج است که اغلب از آن به عنوان توزیع "عادی" یاد می‌شود..

برخی از الگوریتم‌ها مانند رگرسیون خطی و رگرسیون لجستیک صراحتاً فرض می‌کنند که متغیرهای با ارزش واقعی دارای توزیع گاوسی هستند. سایر الگوریتم‌های غیرخطی ممکن است این فرض را نداشته باشند، اما اغلب زمانی که متغیرها دارای توزیع گاوسی هستند بهتر عمل می‌کنند.

این هم برای متغیرهای ورودی با ارزش واقعی در مورد وظایف طبقه‌بندی و رگرسیون و هم برای متغیرهای هدف با ارزش واقعی در مورد وظایف رگرسیون اعمال می‌شود.



شکل 14- اعمال Power Transformaer

برای اجرای power Transformaer هم میتوان به شکل زیر عمل کرد.

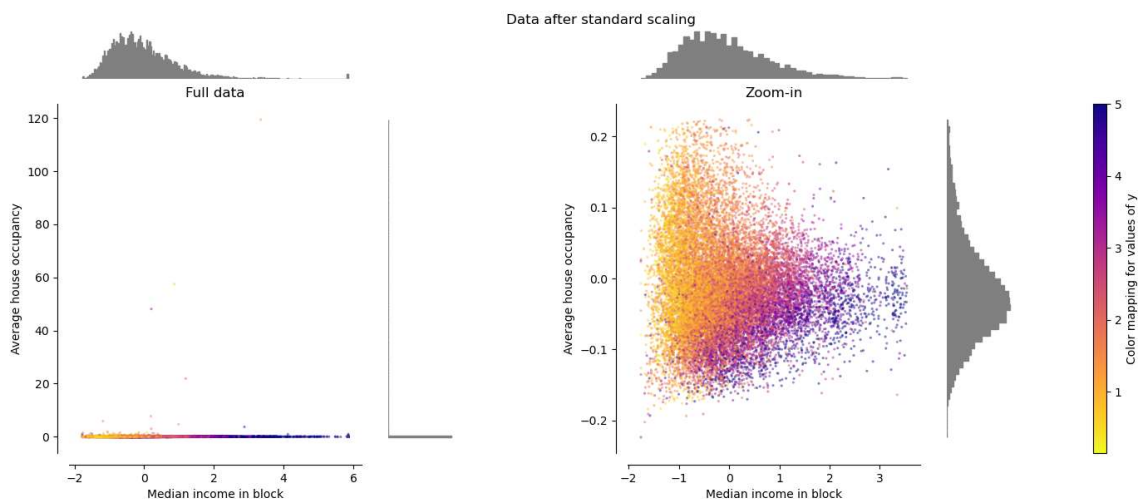
```
## perform a box-cox transform of the dataset
# pt = PowerTransformer()
# data = pt.fit_transform(data)
### convert the array back to a dataframe
```

```
# datasetPT = DataFrame(data)

# datasetPT.columns = ['Area', 'Room', 'Parking', 'Warehouse', 'Elevator']
# datasetPT= pd.concat([datasetPT,dataset['Address']], axis = 1)
# datasetPT= pd.concat([datasetPT,dataset['Price']], axis = 1)
# datasetPT= pd.concat([datasetPT,dataset['Loc']], axis = 1)
# dataset = datasetPT
```

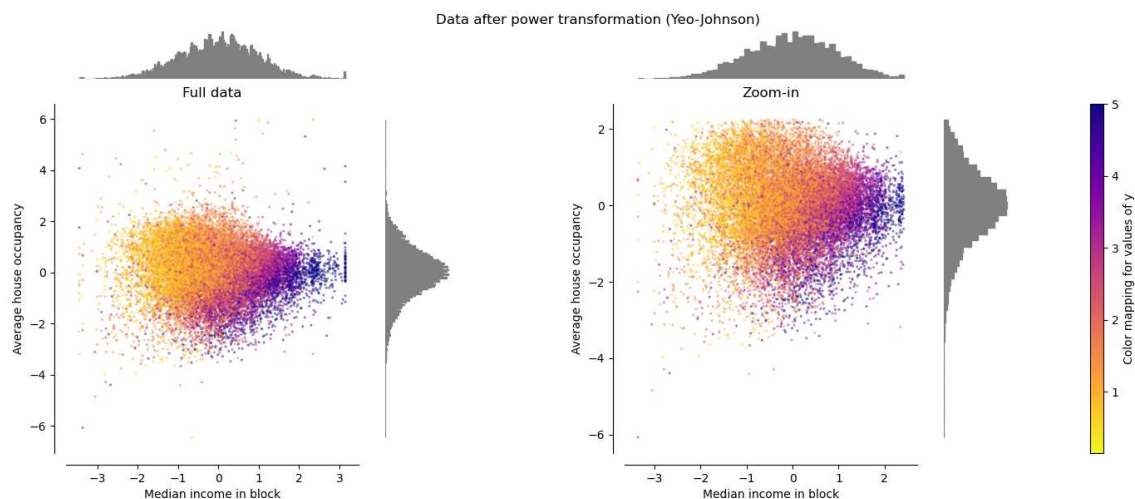
StandardScaler

میانگین را حذف می کند و داده ها را به واریانس واحد مقیاس می دهد. مقیاس بندی محدوده مقادیر ویژگی را همانطور که در شکل سمت چپ زیر نشان داده شده است، کاهش می دهد. با این حال، نقاط پرت هنگام محاسبه میانگین تجربی و انحراف معیار تأثیر دارند. به ویژه توجه داشته باشید که از آنجایی که مقادیر پرت در هر ویژگی دارای مقادیر متفاوتی است، گسترش داده های تبدیل شده در هر ویژگی بسیار متفاوت است: بیشتر داده ها در محدوده $[-2, 4]$ برای ویژگی درآمد متوسط تبدیل شده قرار دارند در حالی که یکسان است. داده ها در محدوده کوچکتر $[-0.2, 0.2]$ برای میانگین اشغال خانه تغییر یافته فشرده می شوند. بنابراین StandardScaler نمی تواند مقیاس ویژگی های متعادل را در حضور نقاط پرت تضمین کند.



شکل 15 - Standard Scaler

در مقابل PowerTransformer برای تثبیت واریانس و به حداقل رساندن چولگی، یک تبدیل قدرت را برای هر ویژگی اعمال می کند تا داده ها را شبیه به گاوسی کند. در حال حاضر تبدیل های Yeo-Johnson و Box-Cox پشتیبانی می شوند و ضریب مقیاس بندی بهینه از طریق تخمین حداکثر احتمال در هر دو روش تعیین می شود. به طور پیش فرض، PowerTransformer عادی سازی واریانس واحد با میانگین صفر را اعمال می کند. توجه داشته باشید که Box-Cox فقط برای داده های کاملاً مثبت قابل اعمال است. درآمد و میانگین اشغال خانه کاملاً مثبت است، اما اگر مقادیر منفی وجود داشته باشد، تبدیل Yeo-Johnson ترجیح داده می شود.



شکل 16- power Transform (منبع سایت sklearn)

در برنامه خصوص زیر StandardScaler را انجام میدهند.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

مرحله سوم - کدگذاری دیتا

Dense Matrix									
1	2	31	2	9	7	34	22	11	5
11	92	4	3	2	2	3	3	2	1
3	9	13	8	21	17	4	2	1	4
8	32	1	2	34	18	7	78	10	7
9	22	3	9	8	71	12	22	17	3
13	21	21	9	2	47	1	81	21	9
21	12	53	12	91	24	81	8	91	2
61	8	33	82	19	87	16	3	1	55
54	4	78	24	18	11	4	2	99	5
13	22	32	42	9	15	9	22	1	21

Sparse Matrix									
1	.	3	.	9	.	3	.	.	.
11	.	4	2	1
.	.	1	.	.	.	4	.	1	.
8	.	.	.	3	1
.	.	.	9	.	.	1	.	17	.
13	21	.	9	2	47	1	81	21	9
.
.	.	.	.	19	8	16	.	.	55
54	4	.	.	.	11
.	.	2	22	.	21

شکل 17- ماتریس Dense , Sparse

ماتریس ها یک عنصر اساسی جبر خطی هستند. ماتریس ها در سراسر زمینه یادگیری ماشین در توصیف الگوریتم ها و فرآیندهایی مانند متغیر داده ورودی (X) هنگام آموزش یک الگوریتم استفاده می شوند.

از آنجایی که ماتریس‌های پراکنده مقادیر زیادی صفر دارند، می‌توانیم الگوریتم‌های خاصی را اعمال کنیم که دو کار مهم را انجام می‌دهند: فشرده‌سازی ردپای حافظه شی ماتریس ما. سرعت بسیاری از روال‌های یادگیری ماشینی

در برنامه خطوط زیر برای HotLabelEncoding استفاده شده است و باعث می‌شود یک ماتریس Spars ایجاد کنیم.

```
address_dummy = pd.get_dummies(house_price['Address'])
house_price_final = house_price.merge(address_dummy, left_index = True, right_index = True)
```

شکل 18- دیتاست بعد از اجرای HotLabelEncoding

همچنین ستونهای Address و Loc را حذف می‌کنیم

```
house_price_final.drop(columns = 'Address', inplace = True)
house_price_final.drop(columns='Loc',inplace=True)
```

Dummy Variable Trap in Regression Models

استفاده از داده‌های طبقه‌بندی شده در مدل‌های رگرسیون چندگانه روشی قدرتمند برای گنجاندن انواع داده‌های غیر عددی در یک مدل رگرسیونی است. داده‌های طبقه‌بندی به مقادیر داده‌ای اشاره دارد که مقوله‌ها را نشان می‌دهند - مقادیر داده‌ای با تعداد ثابت و نامرتب مقادیر، به عنوان مثال جنسیت (مرد/مونت) یا فصل (تابستان/بهار/پاییز). در یک مدل رگرسیونی، این مقادیر را می‌توان با متغیرهای ساختگی - متغیرهایی حاوی مقادیری مانند 1 یا 0 نشان دهنده وجود یا عدم وجود مقدار مقوله‌ای نشان داد. با گنجاندن متغیر ساختگی در یک مدل رگرسیونی، باید مراقب دام متغیر ساختگی بود. دام متغیر ساختگی سناریویی است که در آن متغیرهای مستقل چند خطی هستند - سناریویی که در آن دو یا چند متغیر به شدت همبستگی دارند. به زبان ساده می‌توان یک متغیر را از متغیرهای دیگر پیش بینی کرد

Multi-collinearity

در آمار، چند خطی (همچنین همخطی) پدیده‌ای است که در آن یک متغیر پیش‌بینی‌کننده در یک مدل رگرسیون چندگانه می‌تواند به صورت خطی از بقیه با درجه دقت قابل توجهی پیش‌بینی شود. با توجه به تعاریف بالا یک ستون از X را حذف میکنیم

```
X = house_price_final.drop(columns = 'Price')
y = house_price_final['Price']

##avoid dummy variable trap
X = X.iloc[:, :-1].values
```

مرحله چهارم - تقسیم داده

داده به دو بخش تست و یادگیری تقسیم میشود و میزان تست ابتدا 20 و سپس 30٪ از کل میباشد

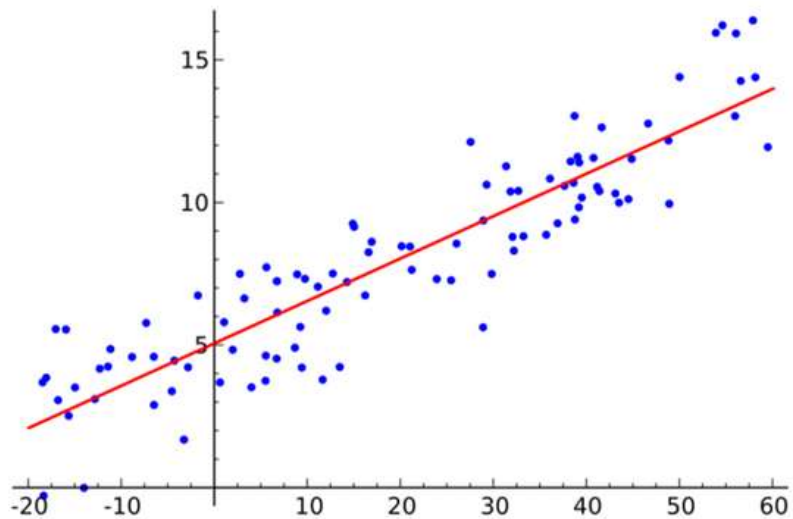
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

مرحله پنجم - الگوریتمها و شرح و مقایسه با پارامتر

• رگرسیون خطی

رگرسیون روشی برای مدل سازی مقدار هدف بر اساس پیش بینی کننده های مستقل است. این روش بیشتر برای پیش بینی و یافتن رابطه علت و معلولی بین متغیرها استفاده می شود. تکنیک های رگرسیون بیشتر بر اساس تعداد متغیرهای مستقل و نوع رابطه بین متغیرهای مستقل و وابسته متفاوت است.

می توان آن را به عنوان روشی برای تجزیه و تحلیل آماری توصیف کرد که می تواند برای مطالعه رابطه بین دو متغیر کمی مورد استفاده قرار گیرد. در درجه اول، با استفاده از روش رگرسیون خطی ساده می توان به دو نکته پی برد: استحکام رابطه بین دو متغیر داده شده. (به عنوان مثال، رابطه بین گرم شدن کره زمین و ذوب شدن یخچال ها) و نکته بعدی مقدار متغیر وابسته در مقدار معینی از متغیر مستقل چقدر است. (به عنوان مثال ، میزان ذوب یخچال طبیعی در سطح مشخصی از گرمایش یا دمای کره زمین



شکل 19- شمای رگرسیون خطی

رگرسیون خطی ساده نوعی تحلیل رگرسیون است که در آن تعداد متغیرهای مستقل یک است و بین متغیر مستقل (X) وابسته (Y) رابطه خطی وجود دارد. خط قرمز در نمودار بالا به عنوان مناسب ترین خط مستقیم معرفی شده است. بر اساس نقاط داده رسم شده، ما سعی می کنیم خطی را ترسیم کنیم که نقاط را به بهترین شکل مدل کند.

○ مزایای رگرسیون خطی

مزایای رگرسیون خطی، پیاده سازی ساده رگرسیون خطی بسیار ساده است که می تواند به راحتی اجرا شود و نتایج رضایت بخشی را ارائه دهد. علاوه بر این، این نمونه مدل ها را می توان به راحتی و به صورت کارآمد حتی در سیستم هایی با قدرت محاسباتی نسبتاً پایین در مقایسه با سایر الگوریتم های پیچیده آموزش داد. رگرسیون خطی، پیچیدگی زمانی کمتری در مقایسه با برخی دیگر از الگوریتم های یادگیری ماشین دارد. معادلات ریاضی رگرسیون خطی نیز به راحتی قابل درک و تفسیر هستند. بنابراین تسلط بر رگرسیون خطی بسیار آسان است. عملکرد رگرسیون خطی در مجموعه های داده جداگانه خطی تقریباً متناسب با مجموعه داده های خطی است و اغلب برای یافتن ماهیت رابطه بین متغیرها استفاده می شود.

در رگرسیون خطی با سازمان دهی داده ها می توانیم کاهش Overfitting داشته باشیم. بیش برآزش یا Overfitting زمانی اتفاق می افتد که مدل آموزشی ما بیش از حد داده ها آموزش را یاد گرفته و نمی تواند مقدار مناسبی برای داده های جدید پیش بینی کند.

○ معایب رگرسیون خطی

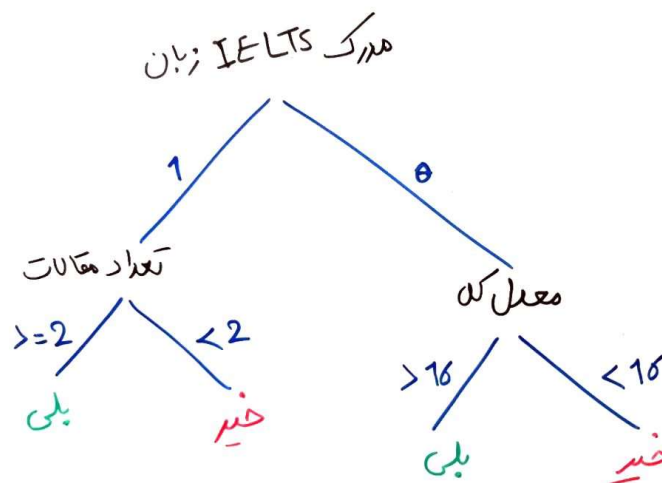
معایب رگرسیون خطی، رگرسیون خطی مستعد Underfitting است: در Underfitting وضعیتی ایجاد می شود که یک مدل یادگیری ماشین نتواند داده ها را به درستی ضبط کند. این معمولاً زمانی رخ می دهد که تابع نمی تواند داده ها را به خوبی سازمان دهی کند. از آنجا که رگرسیون خطی، رابطه خطی بین متغیرهای ورودی و خروجی را در نظر می گیرد، نمی تواند مجموعه داده های پیچیده را به درستی سازمان دهی کند. در اکثر سناریوهای واقعی و داده های موجود، رابطه بین متغیرهای مجموعه داده خطی نیست و بنابراین یک خط مستقیم با داده ها به درستی مطابقت ندارد. در چنین شرایطی یک تابع پیچیده تر می تواند داده ها را به طور موثرتری تحلیل کند. به همین دلیل مدل های رگرسیون خطی دارای دقت پایینی هستند.

این الگوریتم همچنین به داده های پرت شدیداً حساس است؛ این داده ها، داده هایی هستند که از سایر نقاط داده منحرف می شوند. داده های پرت می توانند به عملکرد مدل یادگیری ماشین به شدت آسیب برساند و اغلب می تواند منجر به مدل های کم دقت شوند.

نویز ها و داده های پرت می توانند تأثیر بسیار زیادی بر عملکرد رگرسیون خطی داشته باشند و بنابراین باید قبل از اعمال رگرسیون خطی بر روی مجموعه داده، به طور مناسب مدیریت شوند.

• درخت تصمیم

درخت های طبقه بندی برای طبقه بندی یک مجموعه رکورد به کار می رود. به صورت رایج در فعالیتهای بازاریابی، مهندسی و پزشکی استفاده می شود. در ساختار درخت تصمیم، پیش بینی به دست آمده از درخت در قالب یک سری قواعد توضیح داده می شود. هر مسیر از ریشه تا یک برگ درخت تصمیم، یک قانون را بیان می کند و در نهایت برگ با کلاسی که بیشترین مقدار رکورد در آن تعلق گرفته برچسب می خورد



شکل 20- شمای درخت تصمیم

○ اجزای اصلی درخت تصمیم

برگ (Leaf Nodes): گره هایی که تقسیم های متوالی در آنجا پایان می یابد. برگ ها با یک کلاس مشخص می شوند.

ریشه (Root Node): منظور از ریشه، گره آغازین درخت است.

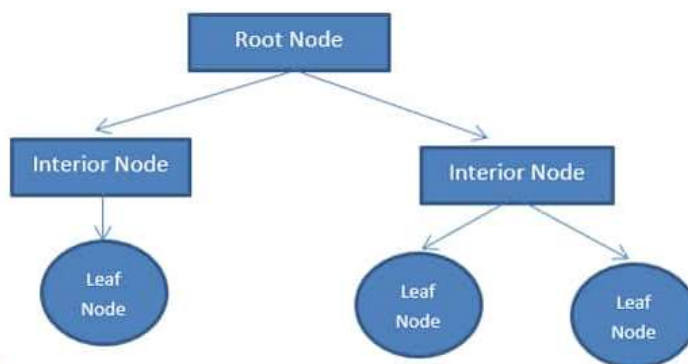
شاخه (Branches): در هر گره داخلی به تعداد جواب های ممکن شاخه ایجاد می شود.

○ معیارهای انتخاب مشخصه برای انشعاب درخت تصمیم

یک معیار انتخاب مشخصه، یک ابتکار برای انتخاب معیار نقطه انشعاب است به طوری که بهترین تفکیک داده های آموزش را به کلاس های برچسب دار داشته باشد.

سه معیار انتخاب مشخصه شناخته شده عبارتند از:

- Information gain
- Gain ratio
- Gini index



شکل 21- نودها در درخت تصمیم

حداقل تعداد نمونه مورد نیاز برای تقسیم یک گره داخلی `min_samples_split` است که یک پارامتر و همچنین پارامتر دیگری با نام `min_samples_leaf` حداقل تعداد نمونه مورد نیاز برای قرار گرفتن در یک گره برگ. است یک نقطه تقسیم در هر عمق تنها در صورتی در نظر گرفته می شود که حداقل نمونه های آموزشی `min_samples_leaf` در هر یک از شاخه های چپ و راست باقی بماند. این ممکن است اثر هموارسازی مدل را به خصوص در رگرسیون داشته باشد.

در کد خطوط زیر برای پیدا کردن مقادیر بهینه اختصاص داده شده است.

```
dtr = DecisionTreeRegressor(random_state = 1)
param_dtr = {'min samples split': [2, 3, 4, 5], 'min samples leaf': [1, 2, 3]}
```

• جنگل تصادفی (Random Forest)

جنگل تصادفی (Random Forest) « یک الگوریتم یادگیری ماشین با قابلیت استفاده آسان است که اغلب اوقات نتایج بسیار خوبی را حتی بدون تنظیم فرآیندهای آن، فراهم می کند. این الگوریتم به دلیل سادگی و قابلیت استفاده، هم برای دسته بندی (Classification) و هم رگرسیون (Regression)، یکی از پر کاربردترین الگوریتم های یادگیری ماشین محسوب می شود

درخت تصمیم، بلوک سازنده جنگل تصادفی

برای درک چگونگی عملکرد جنگل تصادفی، ابتدا باید الگوریتم «درخت تصمیم (Decision Tree)» که بلوک سازنده جنگل تصادفی است را آموخت. انسان ها همه روزه از درخت تصمیم برای تصمیم گیری ها و انتخاب های خود استفاده می کنند، حتی اگر ندانند آنچه که از آن بهره می برند نوعی الگوریتم یادگیری ماشین است. برای شفاف کردن مفهوم الگوریتم درخت تصمیم، از یک مثال روزمره یعنی پیش بینی حداکثر درجه حرارت هوای شهر برای روز بعد (فردا) استفاده می شود.

در اینجا فرض بر آن است که که شهر مورد نظر سیاتل در ایالت واشینگتن واقع شده (این مثال قابل تعمیم به شهرهای گوناگون دیگر نیز هست). برای پاسخ دادن به پرسش ساده «درجه حرارت فردا چقدر است؟»، نیاز به کار کردن روی یک سری از کوئری ها وجود دارد. این کار با ایجاد یک بازه درجه حرارت پیشنهادی اولیه که بر اساس دانش زمینه ای (Domain Knowledge) انتخاب شده، انجام می شود.

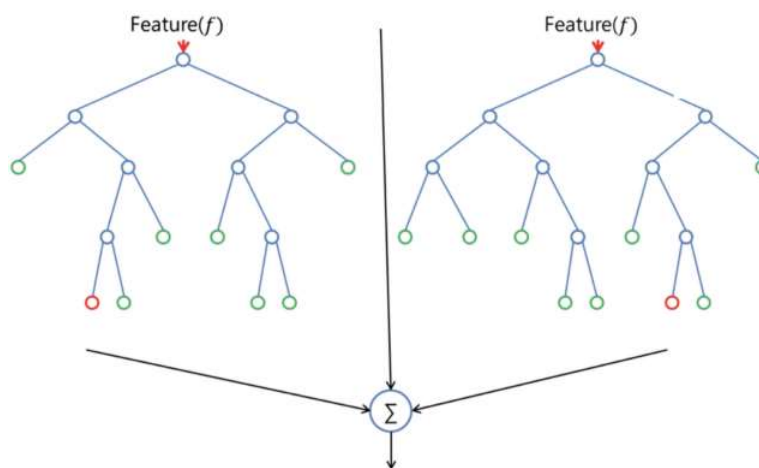
در این مساله چنانچه در آغاز کار مشخص نباشد که «فردا» (که قرار است درجه حرارت آن حدس زده شود) مربوط به چه زمانی از سال است، بازه پیشنهادی اولیه می‌تواند بین ۳۰ الی ۷۰ درجه (فارنهایت) باشد. در ادامه و به تدریج، از طریق یک مجموعه پرسش و پاسخ، این بازه کاهش پیدا می‌کند تا اطمینان حاصل شود که می‌توان یک پیش‌بینی به اندازه کافی مطمئن داشت.

چطور می‌توان پرسش‌های خوبی آماده کرد که پاسخ دادن به آن‌ها بتواند بازه موجود را به شکل صحیحی محدود کند؟ اگر هدف محدود کردن بازه تا حد ممکن باشد، راهکار هوشمندانه آن است که به کوئری‌هایی پرداخته شود که به مساله در حال بررسی مرتبط باشند. از آنجا که درجه حرارت به شدت به زمان سال وابسته است، یک نقطه خوب برای آغاز می‌تواند طرح پرسش «فردا در چه فصلی قرار دارد؟» باشد.

همانطور که مشاهده شد، در یک درخت تصمیم، کار با یک حدس اولیه بر مبنای دانش فرد از جهان آغاز و با دریافت اطلاعات بیشتر، این حدس پالایش می‌شود. طی این فرایند و به تدریج، به گردآوری داده‌ها پایان داده و تصمیمی اتخاذ می‌شود که در اینجا پیش‌بینی بیشینه درجه حرارت است. رویکرد طبیعی که انسان برای حل چنین مساله‌ای مورد استفاده قرار می‌دهد مطابق تصویر ارائه شده در بالا است و می‌توان به آن «روندنمای» (Flowchart) پرسش و پاسخ گفت. در حقیقت، این فلوچارت یک مدل مقدماتی از درخت تصمیم است. اگرچه، در اینجا یک درخت تصمیم کامل ساخته نشده زیرا انسان‌ها از میانبرهایی استفاده می‌کنند که برای ماشین فاقد معنا محسوب می‌شود.

جنگل تصادفی یک الگوریتم یادگیری نظارت شده محسوب می‌شود. همانطور که از نام آن مشهود است، این الگوریتم جنگلی را به طور تصادفی می‌سازد. «جنگل» ساخته شده، در واقع گروهی از «درخت‌های تصمیم» (Decision Trees) است. کار ساخت جنگل با استفاده از درخت‌ها اغلب اوقات به روش «کیسه‌گذاری» (Bagging) انجام می‌شود. ایده اصلی روش کیسه‌گذاری آن است که ترکیبی از مدل‌های یادگیری، نتایج کلی مدل را افزایش می‌دهد. به بیان ساده، جنگل تصادفی چندین درخت تصمیم ساخته و آن‌ها را با یکدیگر ادغام می‌کند تا پیش‌بینی‌های صحیح‌تر و پایدارتری حاصل شوند.

یکی از مزایای جنگل تصادفی قابل استفاده بودن آن، هم برای مسائل دسته‌بندی و هم رگرسیون است که غالب سیستم‌های یادگیری ماشین کنونی را تشکیل می‌دهند. در اینجا، عملکرد جنگل تصادفی برای انجام «دسته‌بندی» (Classification) (تشریح خواهد شد، زیرا گاهی دسته‌بندی را به عنوان بلوک سازنده یادگیری ماشین در نظر می‌گیرند. در تصویر زیر، می‌توان دو جنگل تصادفی ساخته شده از دو درخت را مشاهده کرد.



شکل ۲۲- شمای یک جنگل تصادفی

در کد خطوط زیر برای ایجاد و ست کردن پارامترهای جنگل تصادفی استفاده میشود

```
rfr = RandomForestRegressor(random_state = 1, n_jobs = -1)
param_rfr = {'min_samples_split': [2, 3, 4, 5], 'min_samples_leaf': [1, 2, 3]}
```

• نزدیک ترین همسایگی (k-Nearest Neighbors)

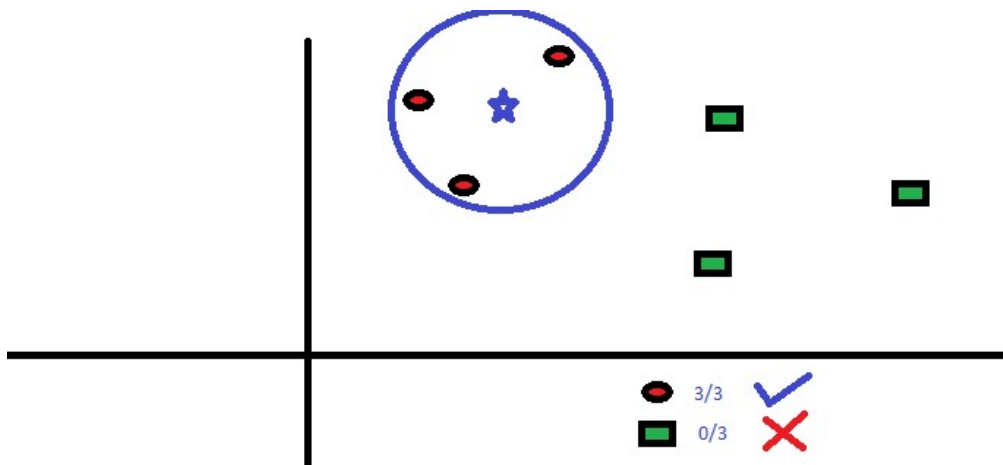
الگوریتم k-نزدیک ترین همسایگی برای مسائل طبقه بندی و رگرسیون قابل استفاده است. اگرچه، در اغلب مواقع از آن برای مسائل طبقه بندی استفاده می شود. برای ارزیابی هر روشی به طور کلی به سه جنبه مهم آن توجه می شود:

1. سهولت تفسیر خروجی ها

2. زمان محاسبه

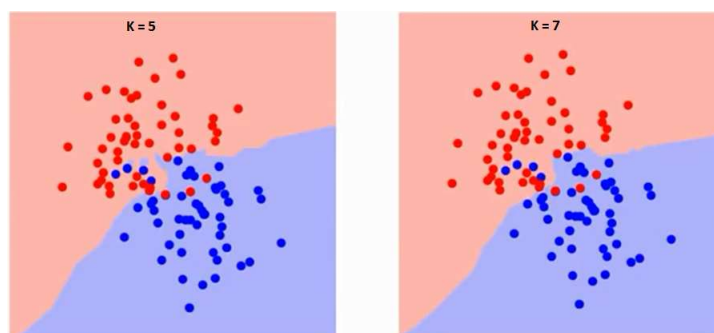
3. قدرت پیش بینی

فرض کنید قصد پیدا کردن کلاسی که ستاره آبی (BS) متعلق به آن است را دارید (یعنی دایره های قرمز). در این مثال در کل دو کلاس دایره های قرمز و مربع های سبز وجود دارند و ستاره آبی بر اساس منطق کلاسیک می تواند تنها متعلق به یکی از این دو کلاس باشد. «K» در الگوریتم k-نزدیک ترین همسایگی، تعداد نزدیک ترین همسایه هایی است که بر اساس آن ها رأی گیری درباره وضعیت تعلق یک نمونه داده به کلاس های موجود انجام می شود. فرض کنید $k = 3$ است



شکل 23- نزدیک ترین همسایه

سه نقطه نزدیک تر به BS، همه دایره های قرمز هستند. با میزان اطمینان خوبی می توان گفت که ستاره آبی به کلاس دایره های قرمز تعلق دارد. در این مثال، انتخاب بسیار آسان است چون هر سه نزدیک ترین همسایه ستاره آبی، دایره های قرمز هستند و در واقع هر سه رأی متعلق به دایره های قرمز است. انتخاب پارامتر k در الگوریتم k-نزدیک ترین همسایگی، بسیار حائز اهمیت است. در ادامه عوامل تأثیرگذار بر انتخاب بهترین k، مورد بررسی قرار می گیرند.



شکل 24- تغییرات K در نزدیک ترین همسایه

همان طور که از تصاویر مشهود است، با افزایش مقدار k ، مرزهای کلاس ها روان تر می شود. با افزایش k به سمت بی نهایت، بسته به اکثریت مطلق نمونه ها در نهایت یک کلاس آبی یا قرمز وجود خواهد داشت.

k-fold Cross-Validation چیست

اعتبارسنجی متقابل یک روش آماری است که برای تخمین مهارت مدل های یادگیری ماشین استفاده می شود. معمولاً در یادگیری ماشین کاربردی برای مقایسه و انتخاب یک مدل برای یک مسئله مدل سازی پیش بینی کننده معین استفاده می شود، زیرا درک آن آسان است، پیاده سازی آن آسان است و منجر به تخمین های مهارتی می شود که عموماً دارای سوگیری کمتری نسبت به روش های دیگر هستند. اعتبارسنجی متقابل یک روش نمونه گیری مجدد است که برای ارزیابی مدل های یادگیری ماشین بر روی یک نمونه داده محدود استفاده می شود. این روش دارای یک پارامتر واحد به نام k است که به تعداد گروه هایی که یک نمونه داده معین قرار است به آنها تقسیم شود، اشاره دارد. به این ترتیب، این روش اغلب اعتبارسنجی متقاطع k -fold نامیده می شود. هنگامی که یک مقدار خاص برای k انتخاب می شود، ممکن است به جای k در مرجع مدل استفاده شود، مانند $k=10$ تبدیل شدن به اعتبار متقاطع 10 برابری.

اعتبارسنجی متقابل اساساً در یادگیری ماشینی کاربردی برای تخمین مهارت مدل یادگیری ماشینی بر روی داده های دیده نشده استفاده می شود. یعنی استفاده از یک نمونه محدود به منظور تخمین نحوه عملکرد کلی مدل زمانی که برای پیش بینی داده های استفاده نشده در طول آموزش مدل استفاده می شود.

این یک روش محبوب است زیرا درک آن ساده است و به دلیل اینکه عموماً منجر به برآوردی کم تر جانبدارانه یا کمتر خوش بینانه از مهارت مدل نسبت به روش های دیگر می شود، روال کلی به شرح زیر است:

مجموعه داده را به صورت تصادفی مخلوط کنید.

مجموعه داده را به k گروه تقسیم کنید

برای هر گروه منحصر به فرد:

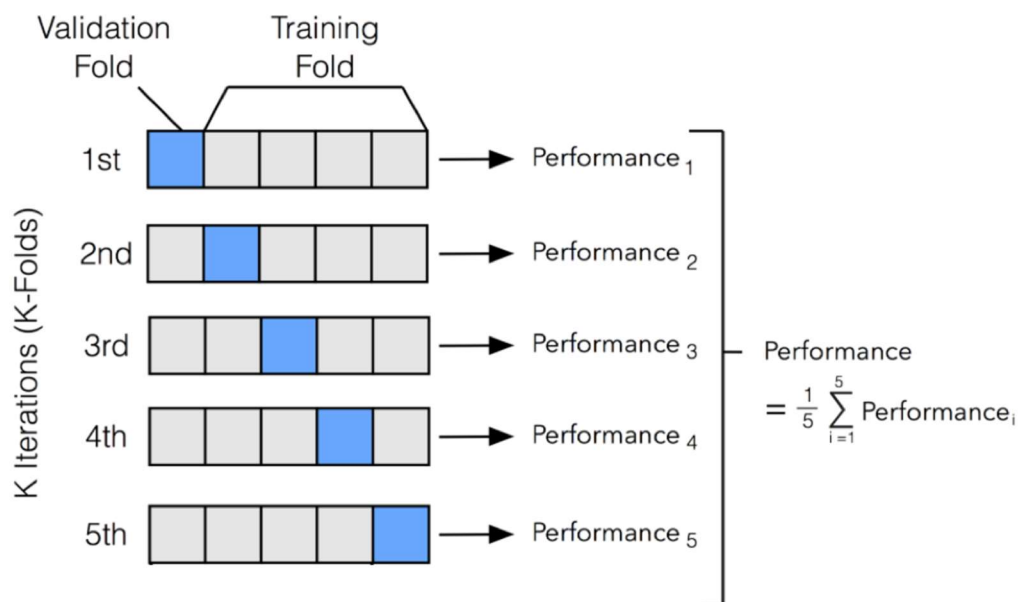
گروه را به عنوان مجموعه داده های نگهدارنده یا آزمایشی در نظر بگیرید

گروه های باقی مانده را به عنوان مجموعه داده های آموزشی در نظر بگیرید

یک مدل را در مجموعه آموزشی قرار دهید و آن را در مجموعه تست ارزیابی کنید

امتیاز ارزیابی را حفظ کنید و مدل را کنار بگذارید

مهارت مدل را با استفاده از نمونه نمرات ارزیابی مدل خلاصه کنید



شکل 25- شمای k-folds

خط زیر در تابع رفتار k-folds را مشخص میکند

```
cv = KFold(shuffle = True, random_state = 1),
```

GridSearchCV

یک تابع کتابخانه ای است که عضوی از بسته model_selection sklearn است. این کمک می کند تا از طریق فرآپارامترهای از پیش تعریف شده حلقه بزنید و برآوردگر (مدل) خود را در مجموعه آموزشی خود قرار دهید. بنابراین، در پایان، می توانید بهترین پارامترها را از لیست شده انتخاب کنید

GridSearchCV یک روش "مناسب" و "امتیاز" را پیاده سازی می کند. همچنین اگر در برآوردگر مورد استفاده اجرا شوند، «score_samples»، «predict»، «predict_proba»، «decision_function»، «transform» و «inverse_transform» را پیاده سازی می کند. پارامترهای تخمین گر مورد استفاده برای اعمال این روش ها با جستجوی شبکه ای اعتبارسنجی متقابل بر روی یک شبکه پارامتر بهینه می شوند.

estimator: شیء برآوردگر که ایجاد کردید

params_grid: شیء دیکشنری که فرآپارامترهایی را که می خواهید امتحان کنید در خود نگه می دارد

scoring: معیار ارزیابی که می خواهید از آن استفاده کنید، می توانید به سادگی یک رشته/بازه معتبر از معیار ارزیابی را ارسال کنید

cv: تعداد اعتبارسنجی متقابلی که باید برای هر مجموعه انتخاب شده از فرآپارامترها امتحان کنید

verbose: می توانید آن را روی 1 تنظیم کنید تا در حالی که داده ها را در GridSearchCV قرار می دهید، پرینت دقیق را دریافت کنید

n_jobs: تعداد فرآیندهایی که می خواهید به صورت موازی برای این کار اجرا شوند، اگر 1- باشد، از تمام پردازنده های موجود استفاده می کند.

```
grid = GridSearchCV(model,
```



```

param_grid = parameters,
refit = True,
cv = KFold(shuffle = True, random_state = 1),
n_jobs = -1)
grid_fit = grid.fit(X_train, y_train)
y_pred = grid_fit.predict(X_test)

```

خطوط بالا جستجو را به شرحی که گفته شد انجام میدهد. همینطور نمایش بهترین نتایج پارامترها به شکل زیر انجام میشود

```
print("The best parameters for {model_name} model is:", grid_fit.best_params )
```

مرحله هفتم - مقایسه نتایج و تفسیر نتایج

در این بخش معیارهای امتیاز دهی مشخص میشود. واضح است که مدل ما بر مبنای رگرسیون است ولی همه معیارها بررسی شده.

Scoring	Function
Classification	
'accuracy'	sklearn.metrics.accuracy_score
'average_precision'	sklearn.metrics.average_precision_score
'f1'	sklearn.metrics.f1_score
'precision'	sklearn.metrics.precision_score
'recall'	sklearn.metrics.recall_score
'roc_auc'	sklearn.metrics.roc_auc_score
Clustering	
'adjusted_rand_score'	sklearn.metrics.adjusted_rand_score
Regression	
'mean_absolute_error'	sklearn.metrics.mean_absolute_error
'mean_squared_error'	sklearn.metrics.mean_squared_error
'r2'	sklearn.metrics.r2_score

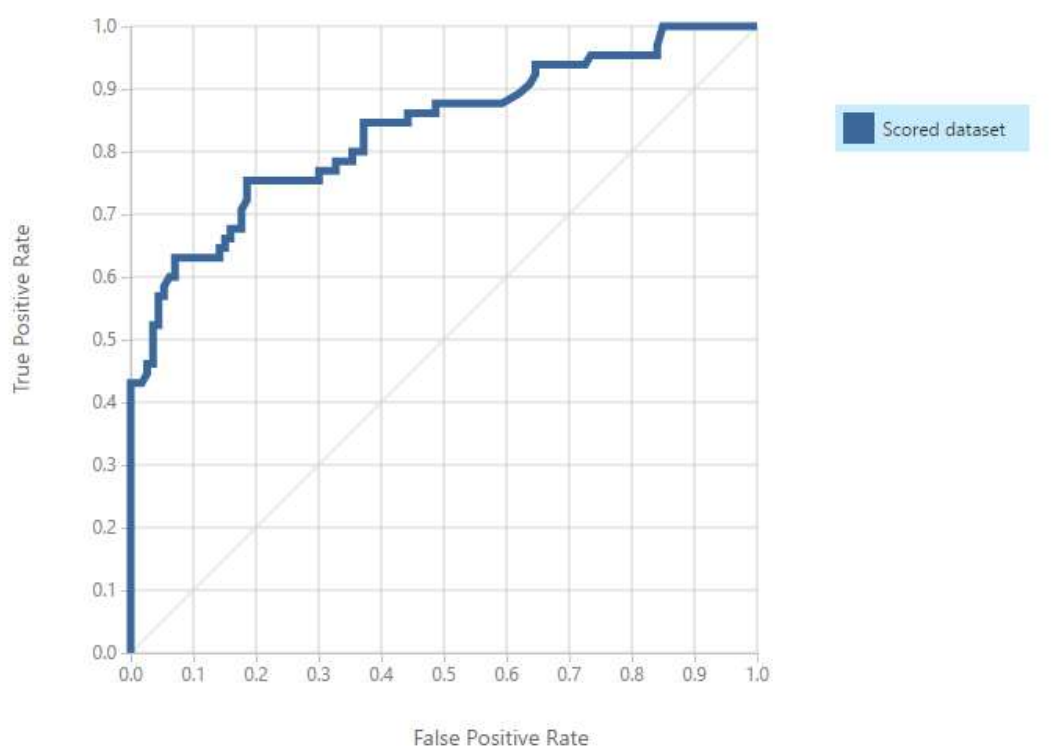
شکل 25- مبانی مقایسه برای کارایی مدل

• Confusion Matrix

ماتریس سردرگمی جدولی است که اغلب برای توصیف عملکرد یک مدل طبقه‌بندی بر روی مجموعه‌ای از داده‌های آزمایشی که مقادیر واقعی آن‌ها مشخص است، استفاده می‌شود. تمام معیارها به جز AUC را می‌توان با استفاده از چهار پارامتر سمت چپ محاسبه کرد. بنابراین، اجازه دهید ابتدا در مورد آن چهار پارامتر صحبت کنیم. مسافرات کشتی تایتانیک را در نظر بگیرید.

Two-Class Boosted Decision Tree > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
41	24	0.803	0.788	0.5	0.835
False Positive	True Negative	Recall	F1 Score		
11	102	0.631	0.701		

شکل 26- نتایج دیتاست تایتانیک

Actual Class	Predicted class	
	Class = Yes	Class = No
	Class = Yes	Class = No
	True Positive	False Negative
	False Positive	True Negative

شکل 27- ماتریس سردرگمی

- True Positives (TP) - اینها مقادیر مثبتی هستند که به درستی پیش بینی شده اند، به این معنی که مقدار کلاس واقعی بله و مقدار کلاس پیش بینی شده نیز بله است. به عنوان مثال. اگر مقدار واقعی کلاس نشان می دهد که این مسافر زنده مانده است و کلاس پیش بینی شده همان چیزی را به شما می گوید.
- منفی های واقعی - (TN) این ها مقادیر منفی پیش بینی شده درست هستند، به این معنی که مقدار کلاس واقعی no و مقدار کلاس پیش بینی شده نیز خیر است. به عنوان مثال. اگر کلاس واقعی می گوید این مسافر زنده نمانده است و کلاس پیش بینی شده همین را به شما می گوید.
- مثبت کاذب و منفی کاذب، این مقادیر زمانی رخ می دهند که کلاس واقعی شما با کلاس پیش بینی شده در تضاد باشد.
- موارد مثبت کاذب - (FP) وقتی کلاس واقعی خیر و کلاس پیش بینی شده بله است. به عنوان مثال. اگر کلاس واقعی بگوید این مسافر زنده نمانده است اما کلاس پیش بینی شده به شما می گوید که این مسافر زنده خواهد ماند.
- منفی های کاذب (FN) - وقتی کلاس واقعی بله است اما کلاس پیش بینی شده در خیر. به عنوان مثال. اگر مقدار واقعی کلاس نشان دهد که این مسافر زنده مانده است و کلاس پیش بینی شده به شما می گوید که مسافر خواهد مرد.
- Accuracy - دقت بصری ترین معیار عملکرد است و صرفاً نسبت مشاهدات پیش بینی شده درست به کل مشاهدات است. ممکن است کسی فکر کند که اگر دقت بالایی داشته باشیم، مدل ما بهترین است. بله، دقت یک معیار عالی است اما فقط زمانی که مجموعه داده های متقارن دارید که مقادیر مثبت کاذب و منفی کاذب تقریباً یکسان هستند. بنابراین، برای ارزیابی عملکرد مدل خود باید به پارامترهای دیگر نگاه کنید. برای مدل ما 0.803 داریم که به این معنی است که مدل ما تقریباً 80 درصد دقیق

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

```
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = [0, 5, 2, 4]
y_true = [0, 1, 2, 3]
accuracy_score(y_true, y_pred)
```

0.5

شکل 28- مثال برای دقت

- **Precision** - دقت نسبت مشاهدات مثبت پیش بینی شده صحیح به کل مشاهدات مثبت پیش بینی شده است. سؤالی که این پاسخ متریک مربوط به همه مسافرانی است که به آنها برچسب زنده مانده اند، واقعاً چند نفر زنده مانده اند؟ دقت بالا به نرخ مثبت کاذب کم مربوط می شود. ما دقت 0.788 را داریم که بسیار خوب است.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

بهترین مقدار 1 و بدترین مقدار 0 است.

- **Recall** - یادآوری نسبت مشاهدات مثبت پیش بینی شده درست به همه مشاهدات در کلاس واقعی است - بله. پاسخ به یادآوری سؤال این است: از بین تمام مسافرانی که واقعاً جان سالم به در بردند، چند نفر را برچسب زدیم؟ ما 0.631 را فراخوانی داریم که برای این مدل خوب است زیرا بالای 0.5 است.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

شکل 29 - دقت و یادآوری

• **F1 score**

امتیاز F1 - امتیاز F1 میانگین وزنی Precision و Recall است. بنابراین، این امتیاز هم مثبت کاذب و هم منفی کاذب را در نظر می گیرد. به طور شهودی درک آن به اندازه دقت آسان نیست، اما F1 معمولاً مفیدتر از دقت است، به خصوص اگر توزیع کلاس ناهمواری داشته باشید. دقت در صورتی بهتر عمل می کند که مثبت کاذب و منفی کاذب هزینه مشابهی داشته باشند. اگر هزینه مثبت کاذب و منفی کاذب بسیار متفاوت است، بهتر است به دقت و یادآوری هر دو نگاه کنید. در مورد ما، امتیاز F1 0.701 است.

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

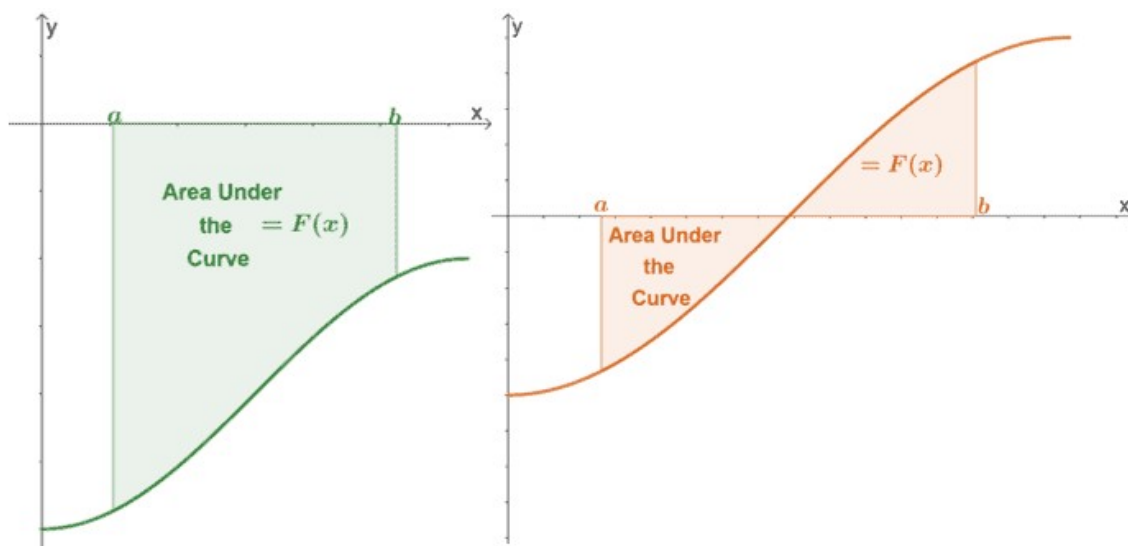
امتیاز F1 به بهترین مقدار خود در 1 و بدترین مقدار در 0 می رسد. امتیاز F1 پایین نشان دهنده دقت ضعیف و یادآوری ضعیف است.

• Average_precision_score

دقت به “دقت در آستانه تصمیم گیری خاص” اشاره دارد. به عنوان مثال، اگر خروجی هر مدلی را کمتر از 0.5 منفی و بیشتر از 0.5 را مثبت حساب کنید. اما گاهی اوقات (به خصوص اگر کلاس های شما متعادل نیستند، یا اگر می خواهید دقت را به یادآوری ترجیح دهید یا برعکس)، ممکن است بخواهید این آستانه را تغییر دهید. دقت متوسط، دقت متوسطی را در تمام این آستانه های ممکن به شما می دهد، که مشابه ناحیه زیر منحنی دقت فراخوان است. این یک معیار مفید برای مقایسه اینکه مدل ها چگونه پیش بینی ها را ترتیب می دهند، بدون در نظر گرفتن هیچ آستانه تصمیم گیری خاصی است.

Area Under the Curve

یکی از مفیدترین کاربردهای حساب انتگرال، یادگیری نحوه محاسبه مساحت زیر منحنی است. انتگرال ها و نواحی معین موجود در زیر منحنی در فیزیک، آمار، مهندسی و سایر زمینه های کاربردی ضروری هستند نواحی زیر منحنی با تابع، دو خط عمودی و محور افقی تشکیل می شوند. مقادیر آنها را می توان با ارزیابی انتگرال معین تابع با توجه به مرزهای عمودی محاسبه کرد.



شکل 30- AUC

منطقه زیر منحنی (AUC) اندازه گیری توانایی طبقه بندی کننده برای تمایز بین کلاس ها است و به عنوان خلاصه منحنی ROC استفاده می شود. هر چه AUC بالاتر باشد، عملکرد مدل در تشخیص کلاس های مثبت و منفی بهتر است. AUC طبقه بندی کننده هایی که عملکرد بدتری نسبت به طبقه بندی کننده های تصادفی دارند. معمولاً AUC در محدوده [0.5,1] است زیرا طبقه بندی کننده های مفید باید بهتر از تصادفی عمل کنند. با این حال، در اصل، AUC همچنین می تواند کوچکتر از 0.5 باشد، که نشان می دهد یک طبقه بندی بدتر از یک طبقه بندی تصادفی عمل می کند.

adjusted_rand_score

شاخص رند با در نظر گرفتن همه جفت‌های نمونه و شمارش جفت‌هایی که در خوشه‌های یکسان یا متفاوت در خوشه‌های پیش‌بینی شده و واقعی تخصیص داده شده‌اند، اندازه‌گیری شباهت بین دو خوشه‌بندی را محاسبه می‌کند. امتیاز رند تعدیل شده برای تعیین شباهت دو نتیجه خوشه‌ای به یکدیگر معرفی شده است. شاخص رند تنظیم شده (ARI) باید به صورت زیر تفسیر شود: $ARI \geq 0.90$ بازبایی عالی. $ARI < 0.90 \geq 0.80$ ریکاوری خوب. $ARI < 0.80 \geq 0.65$ بازبایی متوسط؛ $ARI < 0.65$ بازبایی ضعیف.

• R2

(ضریب تعیین) تابع امتیاز رگرسیون. بهترین امتیاز ممکن 1.0 است و می‌تواند منفی باشد (زیرا مدل می‌تواند خودسرانه بدتر باشد). یک مدل ثابت که همیشه مقدار مورد انتظار y را پیش‌بینی می‌کند، بدون در نظر گرفتن ویژگی‌های ورودی، امتیاز 0.0 دریافت می‌شود. مقدار تغییر در ویژگی وابسته به خروجی است که از متغیر(های) مستقل ورودی قابل پیش‌بینی است. بسته به نسبت انحراف کل نتایج توصیف شده توسط مدل، برای بررسی اینکه چگونه نتایج به خوبی مشاهده شده توسط مدل بازتولید می‌شوند، استفاده می‌شود.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

SS_{res} مجموع مربعات خطاهای باقیمانده است.

SS_{tot} مجموع کل خطاها است.

تفسیر امتیاز R^2 :

$R^2 = 0.68$ را فرض کنید

می‌توان اشاره کرد که 68 درصد از تغییرپذیری ویژگی خروجی وابسته را می‌توان با مدل توضیح داد در حالی که 32 درصد باقیمانده از تغییرپذیری هنوز در نظر گرفته نشده است.

R^2 نسبت نقاط داده را نشان می‌دهد که در خط ایجاد شده توسط معادله رگرسیون قرار دارند. مقدار بالاتر R^2 مطلوب است زیرا نشان دهنده نتایج بهتر است.

```
### Assume y is the actual value and f is the predicted values
```

```
y=[10, 20, 30]
```

```
f=[10, 20, 30]
```

```
r2 = r2_score(y, f)
```

```
print('r2 score for perfect model is', r2)
```

```
r2 score for perfect model is 1.0
```

```
### Assume y is the actual value and f is the predicted values
```

```
y=[10, 20, 30]
```

```
f=[20, 20, 20]
```

```
r2 = r2_score(y, f)
```

```
print('r2 score for a model which predicts mean value always is', r2)
```

```
r2 score for a model which predicts mean value always is 0.0
```

```
### Assume y is the actual value and f is the predicted values
```

```
y = [10, 20, 30]
```

```
f = [30, 10, 20]
```

```
r2 = r2_score(y, f)
```

```
print('r2 score for a worse model is', r2)
```

r2 score for a worse model is -2.0

• MAE Mean Absolute Error

میانگین تفاوت بین مقادیر محاسبه شده و مقادیر واقعی را محاسبه می کند. همچنین به عنوان دقت وابسته به مقیاس نیز شناخته می شود زیرا خطا را در مشاهدات انجام شده در همان مقیاس محاسبه می کند. این به عنوان معیارهای ارزیابی برای مدل های رگرسیون در یادگیری ماشین استفاده می شود

در هر صورت، هر چه مقدار MAE به 0 نزدیکتر باشد، بهتر است. همانطور که گفته شد، تفسیر MAE کاملاً به داده ها بستگی دارد. در برخی موارد، MAE 10 می تواند فوق العاده خوب باشد، در حالی که در برخی دیگر می تواند به این معنی باشد که مدل یک شکست کامل است.

تفسیر MAE به موارد زیر بستگی دارد:

محدوده مقادیر،

قابل قبول بودن خطا

```
import numpy as np
from sklearn.metrics import mean_absolute_error
```

```
true = [1,2,3,4,5,6]
predicted = [1,3,4,4,5,9]
```

```
print(mean_absolute_error(true, predicted))
```

```
# Returns: 0.833
```

• MAPE-Mean Absolute Percentage Error

میانگین درصد خطای مطلق (MAPE) را می توان در یادگیری ماشین برای اندازه گیری دقت یک مدل استفاده کرد. به طور خاص، MAPE یک تابع ضرر است که خطای یک مدل معین را تعریف می کند.

MAPE با یافتن تفاوت مطلق بین مقادیر واقعی و پیش بینی شده، تقسیم بر مقدار واقعی محاسبه می شود. این نسبت ها برای همه مقادیر اضافه شده و میانگین گرفته می شود.

MAPE یک معیار رایج در یادگیری ماشینی است زیرا تفسیر آن آسان است. هرچه مقدار MAPE کمتر باشد، مدل یادگیری ماشینی در پیش بینی مقادیر بهتر عمل می کند. برعکس، هرچه مقدار MAPE بیشتر باشد، مدل در پیش بینی مقادیر بدتر است.

به عنوان مثال، اگر مقدار MAPE 20% را برای یک مدل یادگیری ماشینی معین محاسبه کنیم، میانگین تفاوت بین مقدار پیش بینی شده و مقدار واقعی 20٪ خواهد بود.

به عنوان درصد، اندازه گیری خطا نسبت به معیارهای دیگر مانند میانگین مربعات خطا، شهودی تر است. این به این دلیل است که بسیاری از اندازه گیری های خطای دیگر نسبت به محدوده مقادیر هستند. این امر مستلزم آن است که از برخی موانع ذهنی اضافی عبور کنید تا دامنه خطا را تعیین کنید.

> 50 %	Poor
20% – 50%	Relatively good
10% – 20%	Good
< 10%	Great

```
# A practical example of MAPE in sklearn
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_percentage_error

data = load_diabetes()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(X, y)

lnr = LinearRegression()
lnr.fit(X_train, y_train)
predictions = lnr.predict(X_test)

print(mean_absolute_percentage_error(y_test, predictions))

# Returns: 0.339
```

• Root Mean Square Error -RSME

خطای مربع میانگین یکی از این معیارهای خطا برای قضاوت در مورد دقت و میزان خطای هر الگوریتم یادگیری ماشینی برای مسئله رگرسیون است.

بنابراین، MSE یک تابع ریسک است که به ما کمک می کند میانگین مجذور اختلاف بین مقدار پیش بینی شده و واقعی یک ویژگی یا متغیر را تعیین کنیم. RMSE مخفف Root Mean Square Error است که جذر مقدار بدست آمده از تابع Mean Square Error است. با استفاده از RMSE، به راحتی می توانیم تفاوت بین مقادیر تخمینی و واقعی یک پارامتر از مدل را رسم کنیم. با این کار، به وضوح می توان کارایی مدل را قضاوت کرد. بر اساس یک قانون سرانگشتی، می توان گفت که مقادیر RMSE بین 0.2 و 0.5 نشان می دهد که مدل می تواند داده ها را به طور نسبی با دقت پیش بینی کند. علاوه بر این، Adjusted R-squared بیشتر از 0.75 مقدار بسیار خوبی برای نشان دادن دقت است. در برخی موارد، Adjusted R-squared 0.4 یا بیشتر نیز قابل قبول است.

```
from sklearn.metrics import mean_squared_error
import math
y_actual = [1,2,3,4,5]
```



```
y_predicted = [1.6,2.5,2.9,3,4.1]
```

```
MSE = mean_squared_error(y_actual, y_predicted)
```

```
RMSE = math.sqrt(MSE)
print("Root Mean Square Error:\n")
print(RMSE)
```

```
0.6971370023173351
```

در برنامه در خطوط زیر مقادیر خطا قابل مشاهده میشود.

```
train_score = round(grid_fit.score(X_train, y_train), 4)
test_score = round(grid_fit.score(X_test, y_test), 4)
RMSE = metrics.mean_squared_error(y_test, y_pred, squared=False) #Root Mean Squared Error
MAE = metrics.mean_absolute_error(y_test, y_pred) #Mean Absolute Error (MAE)
MAPE = metrics.mean_absolute_percentage_error(y_test, y_pred) #Mean Absolute Percentage Error
```

Bias–variance tradeoff

در آمار و یادگیری ماشین، مبادله بایاس-واریانس ویژگی مدلی است که واریانس پارامتر تخمین زده شده در بین نمونه ها را می توان با افزایش بایاس در پارامترهای تخمین زده کاهش داد.

هدف هر الگوریتم یادگیری ماشینی نظارت شده این است که تابع نگاشت (f) را برای متغیر خروجی (Y) با توجه به داده های ورودی (X) به بهترین شکل تخمین بزند. تابع نگاشت اغلب تابع هدف نامیده می شود، زیرا تابعی است که الگوریتم یادگیری ماشینی تحت نظارت معین قصد تقریب آن را دارد. خطای پیش بینی هر الگوریتم یادگیری ماشینی را می توان به سه بخش تقسیم کرد:

خطای سوگیری

خطای واریانس

خطای کاهش ناپذیر

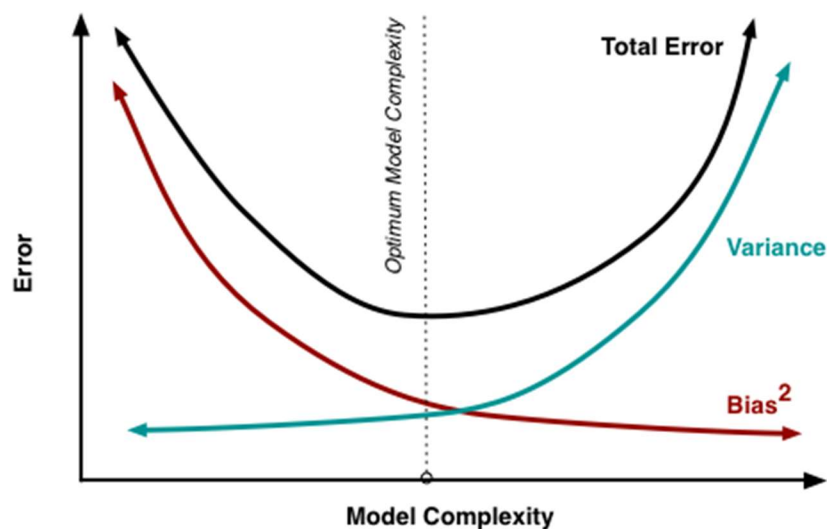
خطای کاهش ناپذیر را نمی توان صرف نظر از اینکه از چه الگوریتمی استفاده می شود کاهش داد. این خطای وارد شده از چارچوب انتخابی مسئله است و ممکن است توسط عواملی مانند متغیرهای ناشناخته ایجاد شود که بر نگاشت متغیرهای ورودی به متغیر خروجی تأثیر می گذارد.

• Bias Error

بایاس تفاوت بین پیش بینی میانگین مدل ما و مقدار درستی است که ما سعی داریم آن را پیش بینی کنیم. مدل با بایاس بالا توجه بسیار کمی به داده های آموزشی دارد و مدل را بیش از حد ساده می کند. این اتفاق همیشه منجر به خطای بالا در داده های آموزشی و تستی می شود.

• Variance Error

واریانس تغییرپذیری پیش بینی مدل برای یک نقطه داده مشخص یا مقداری است که به ما میزان گستردگی داده ها را می گوید. مدل با واریانس بالا توجه زیادی به داده های آموزشی می کند و برای داده هایی که قبلاً ندیده است تعمیم ایجاد نمی کند. در نتیجه، چنین مدل هایی در داده های آموزشی بسیار خوب عمل می کنند اما نرخ خطای بالایی در داده های تست دارند.



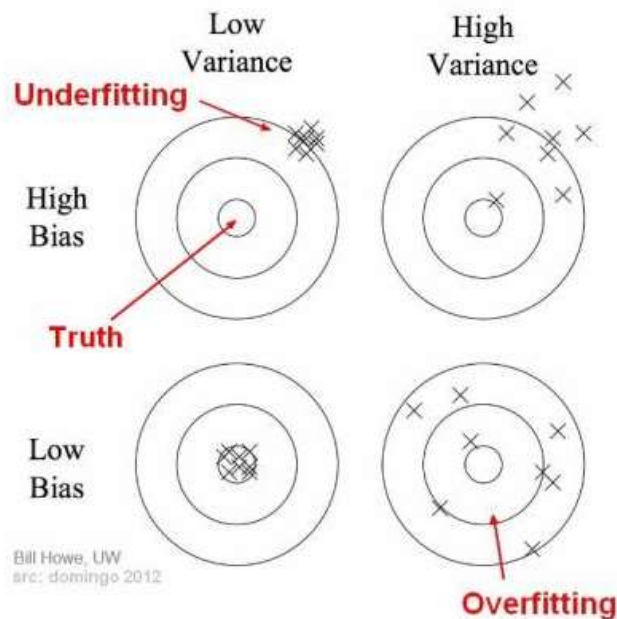
شکل 31 - معادله Bias-variance

مسئله اصلی آن است که در یک مدل مناسب، هم بایاس و هم واریانس باید حداقل ممکن باشند. ولی متأسفانه، کمینه‌سازی (Minimization) هر دو این شاخص‌ها به شکل توأم، امکان‌پذیر نیست. چنین وضعیتی را «تناقض واریانس-اریبی» (Bias-Variance Dilemma) می‌نامند

• Overfitting vs. Underfitting

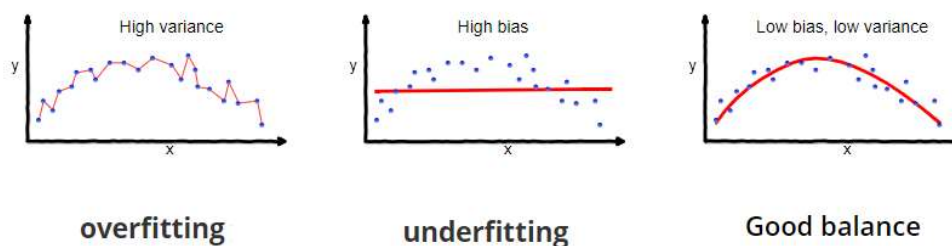
تطبیق بیش از حد: عملکرد خوب در داده‌های آموزشی، تعمیم ضعیف به داده‌های دیگر. عدم تناسب: عملکرد ضعیف در داده‌های آموزشی و تعمیم ضعیف به داده‌های دیگر

در یادگیری تحت نظارت، عدم تناسب زمانی اتفاق می‌افتد که یک مدل قادر به درک الگوی اساسی داده‌ها نباشد. این مدل‌ها معمولاً دارای بایاس بالا و واریانس پایین هستند. این اتفاق زمانی می‌افتد که ما مقدار داده بسیار کمتری برای ساخت یک مدل دقیق داریم یا زمانی که سعی می‌کنیم یک مدل خطی با داده‌های غیرخطی بسازیم. همچنین، این نوع مدل‌ها برای بدست آوردن الگوهای پیچیده در داده‌هایی مانند رگرسیون خطی و منطقی بسیار ساده هستند. در یادگیری تحت نظارت، بیش برآزش زمانی اتفاق می‌افتد که مدل ما نویز را همراه با الگوی اساسی در داده‌ها بگیرد. این موضوع زمانی اتفاق می‌افتد که ما مدل خود را بر روی مجموعه داده با نویز آموزش می‌دهیم. این مدل‌ها دارای بایاس پایین و واریانس بالا هستند. چنین مدل‌هایی بسیار پیچیده‌اند؛ مانند درخت‌های تصمیم‌گیری که مستعد بیش برآزش هستند.



شکل 32- بایاس و واریانس با استفاده از دیاگرام چشم گاوی

در نمودار بالا، مرکز هدف مدلی است که مقادیر صحیح را به خوبی پیش‌بینی می‌کند. هرچه از نقطه مرکزی دور می‌شویم، پیش‌بینی‌مان بدتر و بدتر می‌شود. ما می‌توانیم فرآیند ساخت مدل خود را برای گرفتن ضربات جداگانه بر روی هدف تکرار کنیم. در یادگیری تحت نظارت، عدم تناسب زمانی اتفاق می‌افتد که یک مدل قادر به درک الگوی اساسی داده‌ها نباشد. این مدل‌ها معمولاً دارای بایاس بالا و واریانس پایین هستند. این اتفاق زمانی می‌افتد که ما مقدار داده بسیار کمتری برای ساخت یک مدل دقیق داریم یا زمانی که سعی می‌کنیم یک مدل خطی با داده‌های غیرخطی بسازیم. همچنین، این نوع مدل‌ها برای بدست آوردن الگوهای پیچیده در داده‌هایی مانند رگرسیون خطی و منطقی بسیار ساده هستند. در یادگیری تحت نظارت، بیش‌برازش زمانی اتفاق می‌افتد که مدل ما نویز را همراه با الگوی اساسی در داده‌ها بگیرد. این موضوع زمانی اتفاق می‌افتد که ما مدل خود را بر روی مجموعه داده با نویز آموزش می‌دهیم. این مدل‌ها دارای بایاس پایین و واریانس بالا هستند. چنین مدل‌هایی بسیار پیچیده‌اند؛ مانند درخت‌های تصمیم‌گیری که مستعد بیش‌برازش هستند.



$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

شکل 33-مجموع خطا

مدلی که واریانس کوچک و بایاس زیاد را نشان می‌دهد، کمتر به هدف می‌رسد، در حالی که مدلی با واریانس بالا و بایاس کم، بیش از حد با هدف مطابقت دارد.

Index	Training score ▲	Testing score	RMSE	MAE	MAPE
LinearRegression	0.691900	-86395975996155...	214025698395860172...	118537871272579302...	408751280.262803
RandomForestRegressor	0.943300	0.775900	3447296189.487419	1692304844.589384	0.291986
DecisionTreeRegressor	0.968800	0.525900	5013723269.523179	2138199926.964651	0.339366
KNeighborsRegressor	0.996000	0.770900	3485165375.989917	1691111943.393756	0.305812

شکل 34- اجرا با 20 درصد تست

Index	Training score	Testing score	RMSE	MAE	MAPE
LinearRegression	0.764100	-1187753900...	2578134424016931782656.000000	181931247337884123136.000000	61633350337.476578
DecisionTreeRegressor	0.970900	0.566500	4925523096.330745	2010413720.907586	0.295814
RandomForestRegressor	0.943200	0.791100	3419491071.418974	1621531119.172595	0.262401
KNeighborsRegressor	0.817100	0.747000	3762412066.565140	1770857975.460123	0.332249

شکل 35- اجرا با 30 درصد تست

همچنان که مشاهده میشود تفاوت زیادی بین امتیاز تست و آموزش وجود دارد و این مقدار به خاطر نداشتن دیتای کافی ایجاد شده است. همینطور درصد خطا حدود 30 میباشد که این مشکلاتی را در پی دارد این وضعیت متعادل نیست.

مرحله هشتم – افزایش نمونه ها در صورت نیاز

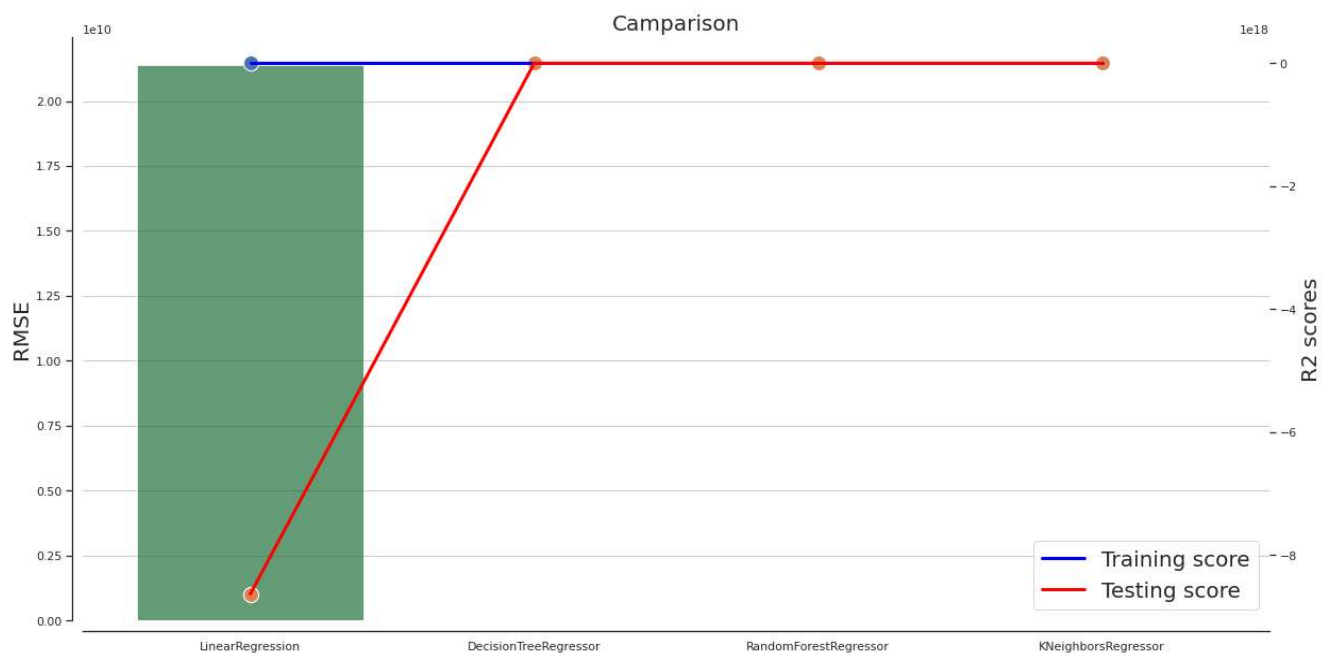
با انتخاب نمونه‌ها به صورت تصادفی و جایگزینی، به نمونه‌برداری بیش از حد از کلاس(های) اقلیت اعتراض کنید. بوت استرپ را می‌توان به صورت صاف تولید کرد. نمونه برداری تصادفی با استفاده از کلاس RandomOverSampler قابل پیاده سازی است. کلاس را می‌توان تعریف کرد و یک آرگومان sampling_strategy را می‌گیرد که می‌تواند روی "اقلیت" تنظیم شود تا به طور خودکار کلاس اقلیت را با کلاس یا کلاس های اکثریت متعادل کند. خطوط زیر برای ایجاد نمونه های کافی نوشته شده است.

```
from imblearn.over_sampling import RandomOverSampler

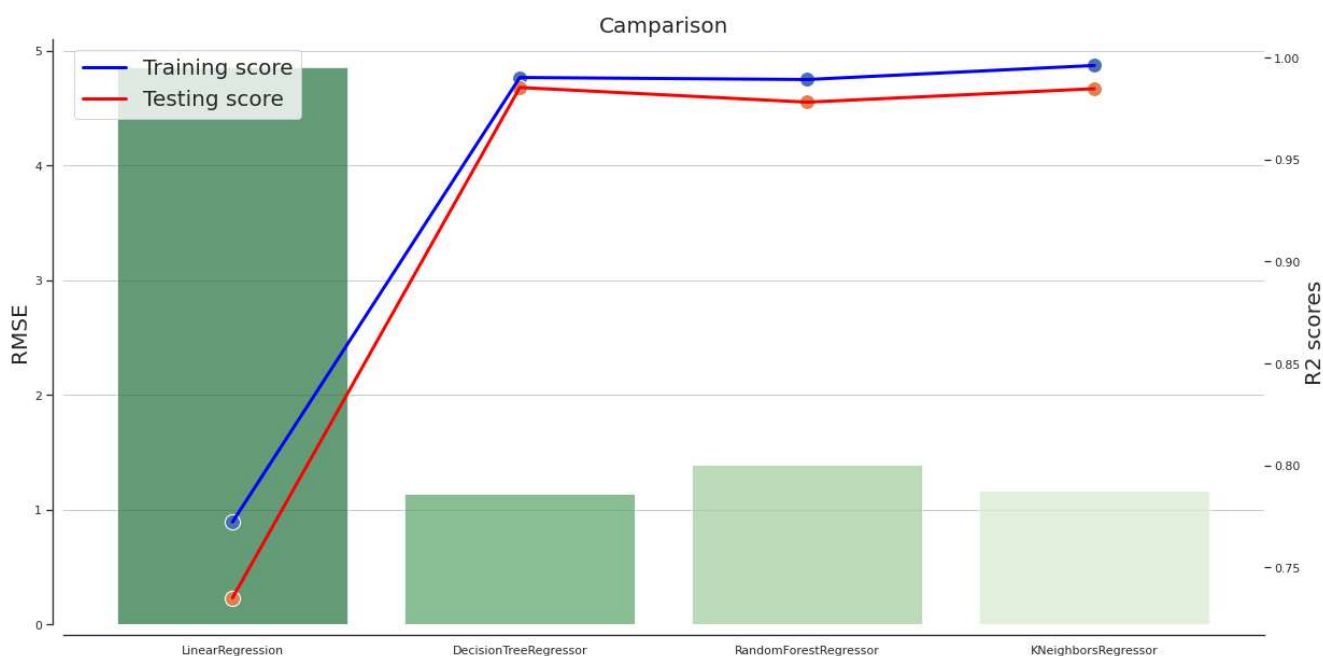
oversample = RandomOverSampler(sampling_strategy='auto') #oversampling class
X_over, y_over = oversample.fit_resample(X, y)
X, y = oversample.fit_resample(X, y)
```

Index	Training score	Testing score	RMSE	MAE	MAPE
LinearRegression	0.772300	0.735100	4861542895.344919	2636673470.361132	0.697947
RandomForestRegressor	0.989300	0.978200	1393668637.795958	316048808.022576	0.057944
KNeighborsRegressor	0.996200	0.984800	1162627582.127394	245523431.082507	0.040621
DecisionTreeRegressor	0.990300	0.985400	1141881938.306298	258129211.082732	0.044904

شکل 36- اجرا با 30 درصد تست و oversampler



شکل 37- اجرا با 30 درصد تست



شکل 38- اجرا با 30 درصد تست و oversampler

سخن پایانی

مراحل کلی برای انجام فرایند داده کاوی توسط python مرحله به مرحله انجام شد. Sklearn و متد GridSearch به صورت خودکار پارامترهایی که مشخص شده باشند را تست میکنند. باید توجه داشت که مدل ما رگرسیون میباشد. در نهایت با ایجاد تعدادی نمونه اضافی مدل به شکل متعادل با درصد خطای حدود 4٪ در الگوریتمهای درخت تصمیم و نزدیک ترین همسایه قابل استفاده است.

مراجع

- [1] M. K. J. P. Jiawei Han, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2011.
- [2] J. Dean, Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners, Wiley, 2014.
- [3] V. K. M. S. Pang-Ning Tan, Introduction to Data Mining (Second Edition), 2005.
- [4] R. Layton, Learning Data Mining with Python, Packt Publishing, 2015.
- [5] S. Raschka, Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition, PacktPublishing, 2019.