آبتین ماه یار – ۹۸۲۲۲۰۸۷

متد های Forward and backward selection در فایل utils.py در قالب یک تابع واحد sequential_feature_selector پیاده سازی شده اند که میتوان با تغییر مقدار direction مدل selection را انتخاب کرد. توضیحات بیشتر در مورد پارامتر های تابع داخل کد داده شده است.

بعد از خواندن دیتا و convert کردن dtypes های هر فیچر به مقدار مناسب آن، موبایل های با price range ۰ و ۱ به ۰ و موبایل های ۲ و ۳ به مقدار ۱ طبق سند تمرین تغییر پیدا کردند. سپس دیتای تست و ترین از دیتاست اصلی استخراج شد. سپس توسط تابع پیاده سازی شده در این بخش ۵ فیچر برتر از بین بقیه فیچر ها با روش های forward و backward انتخاب شدند که هر دو روش فیچر های یکسانی را انتخاب کرده بودند که در ادامه لیست شده اند.

'ram', 'battery_power', 'px_width', 'px_height', 'mobile_wt'

با استفاده از مدل لجستیک با پارامتر های دیفالت و فیچر های انتخابی مدلی طراحی کردیم که نتایج آن در زیر قابل مشاهده است. نتایج بر روی دیتای تست بدست آمده اند.

	precision	recall	f1-score	support
0	0.9851	0.9950	0.9900	200
1	0.9949	0.9850	0.9899	200
accuracy			0.9900	400
macro avg	0.9900	0.9900	0.9900	400
weighted avg	0.9900	0.9900	0.9900	400

در مرحله ی بعدی بر روی دیتای اصلی pca اعمال کردیم تا ۵ تا از بهترین فیچر هایی که واریانس بیشتری از داده را دارند و میتوانند دیتا را به صورت کلی بهتر و در dimension کمتری خلاصه کنند انتخاب کردیم و با استفاده از مدل لجستیک با پارامتر های دیفالت آزمایش بالا را تکرار کردیم که نتایج در زیر آمده است.

	precision	recall	f1-score	support
0	0.9851	0.9950	0.9900	200
1	0.9949	0.9850	0.9899	200
accuracy			0.9900	400
macro avg	0.9900	0.9900	0.9900	400
weighted avg	0.9900	0.9900	0.9900	400

در بخش مهندسی ویژگی ۵ حالت مختلف را پیاده سازی کردیم و با یک مدل SVM کلاسیفایر با پارامتر های دیفالت ترین کردیم که نتیجه هر آزمایش به علاوه آزمایش دیتاست اولیه بدون تغییر در زیر آمده است.

	Dataset	Train f1 score	Test f1 score	Train precision	Test precision	Train recall	Test recall
0	Base dataset (without feature engineering)	0.991875	0.959991	0.991894	0.960414	0.991875	0.96
1	Battery power feature binnded into 3 bins	0.990625	0.952497	0.990632	0.952602	0.990625	0.9525
2	One hot encoding for n_cores feature	0.99	0.964996	0.990012	0.965186	0.99	0.965
3	Transformed skewed distributions	0.990625	0.957487	0.990644	0.958061	0.990625	0.9575
4	Extracted features	0.983125	0.947492	0.983144	0.94778	0.983125	0.9475
5	Mixture of all of the above	0.985625	0.944978	0.985686	0.945713	0.985625	0.945

Third dataset (One hot encoding for n_cores feature) has the best results among the others for the svm classifier.

* از روش one hot encoding تنها در زمانی باید استفاده کرد که اولا دیتای ما nominal باشد (ترتیب خاصی نداشته باشند) و سپس کاردینالیتی آن کمتر از تقریبا ۱۵ تا باشد (بستگی به خود دیتا دارد و حد مشخصی ندارد) زیرا اگر بیشتر باشد به حدی dimensionality کل دیتا را زیاد میکند که باعث میشود عملکرد بیشتر مدل های موجود بد تر شود (curse of dimensionality) و همچنین باعث sparse شدن بیش از حد دیتا میشود. اگر این شرایط را داشته باشیم میتوان از این روش با تبدیل دیتای کتگوریکال به وکتور های با معنی بهترین استفاده را کرد تا عملکرد مدل را افزایش داد. مزیت دیگری که این روش دارد این است که از بایاس شدن دیتا به یک کتگوری جلوگیری می کند.

* اولا اعمال یک transformer بر روی یک داده با skewed distribution در مرحله preprocessing لازم است زیرا مقدار زیادی از مدل های معروف این فرض را بر روی دیتا ها دارند که دیتا ها دارای normal distribution هستند ممکن است دیتا هایی که در قسمت پایینی و کشیده شده ی توزیع دیتا را به عنوان outlier به حساب بیاورند و بر روی مدل هایی که به outlier هستند نیز تاثیر بگذارند، پس با این حرکت میتوانیم دیتا را بر روی مدل های آماری بیشتری فیت کنیم و احتمال خطا را کاهش دهیم. حال transformer های مختلفی وجود دارند که تعدادی از آن ها در زیر آمده اند:

-log transformer : که مقداری به دیتای اوتلایر حساس است و نقطه ضعف آن این است که این دیتا را فقط بر روی دیتای مثبت (غیر از صفر و منفی) میتوانیم اعمال کنیم و عموما برای نرمال کردن توزیع های با چولگی راست استفاده میشود، که از آنجایی که دیتا های skewed ما دارای مقدار صفر در خودشان بودند از این transformer استفاده نشد. (هر چند میتوانستیم با افزودن مقدار کمی به تمام رکورد ها به دیتاهایی که شامل صفر بودند از این ترنسفورمر استفاده کنیم.)

-power transformer: There are two options for Power transformation: Yeo-Johnson transform and Box-Cox transform. However, Box-Cox can only be applied to strictly positive data. both the Yeo-Johnson and Box-Cox method standardize the data to resemble a normal distribution. However, these methods don't scale the data to a

predetermined range. Yeo-Johnson transformation is also having the ability to make the distribution more symmetric.

–square root : که بر روی دیتای مثبت و صفر میتواند اعمال شود. بنابراین این روش را بر روی دیتا هایی که چولگی بیشتری داشتند اعمال کردیم و سپس با استفاده از MinMaxScaler تمام دیتا ها را اسکیل کردیم تا چولگی تمام دیتا ها کاهش یابد و توزیع داده نزدیک به توزیع نرمال شود.

cross validation and bootstrapping:

هر دو این ها متد های resampling هستند به این صورت که ما دیتاست جدیدی از دیتاست اصلی می سازیم و به صورت رندوم از دیتاست اصلی رکورد هایی را انتخاب می کنیم تفاوتی که بین این دو وجود دارد این است که در bootstrapping ما می توانیم یک رکورد را چندین بار انتخاب کنیم و یا اصلا رکوردی در دیتای اصلی را هیچ وقت انتخاب نکنیم تا دیتاستی به اندازه ی دیتاست اصلی بسازیم ولی در cross validation ما چندین دیتاست کوچکتر از دیتای اصلی میسازیم که هر رکورد دقیقا در یکی از این دیتاست ها قرار دارد. این روش چندین حالت مختلف دارد از جمله k-fold cv (دیتا را k بخش تقسیم می کنیم و هر بخش دقیقا یک بار به عنوان دیتای تست استفاده خواهد شد که به leave-x-out cv نیز معروف است. همچنین leave-one-out cv نیز داریم که در انجا صرفا یک ریکورد در هر سری خارج از دیتای اصلی قرار میگیرد)، stratified cv، Time series CV.

به خاطر این که تمام دیتا داخل روش cv استفاده میشود و میتوان یک مدل را چندین بار با دیتاست های کوچک تر ترین و تست کرد بیش تر از این روش برای model performance evaluation استفاده میشود و از روش bootstrapping برای ساختن مدل های ensemble که نتایج چندین مدل را با هم تجمیع و اعلام میکنند و یا تخمین زدن پارامتر های آماری از روی یک مشاهده برای کل جامعه استفاده میشود برای مثال میانگین یا میانه یک متغییر داخل یک جامعه توسط یک مشاهده که با استفاده از آن میتوان تخمینی از میانگین و میانه کل جامعه با یک confidence intervalاعلام کرد.

۵*۲ Cross Validation :

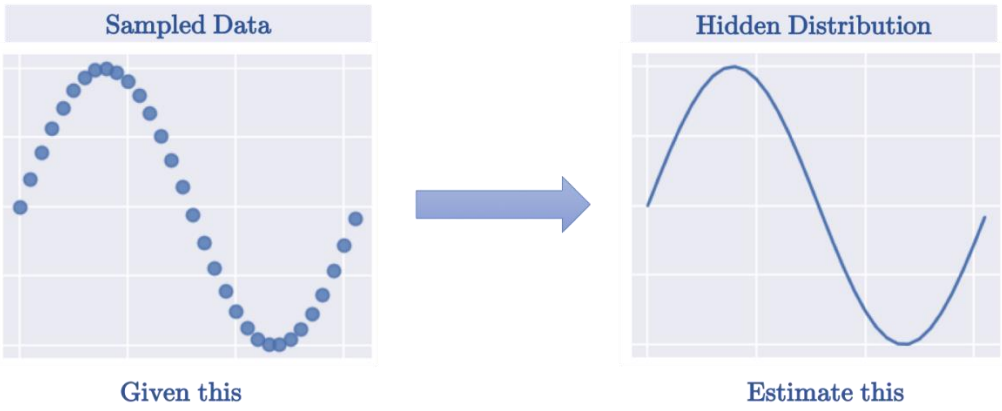
در این متد از همان روش K-Fold CV استفاده میشود که k در این جا مقدار ۲ را دارد یعنی دیتاست اصلی به ۲ بخش مساوی تقسیم شده است و این حرکت ۵ بار متوالی تکرار میشود و میانگین نتایج به عنوان نتیجه ی اصلی اعلام می شود. که از این متد میتوان برای تخمین خطای بوجود آمده توسط مدل های پیش بینی و واریانس آن خطاها برای اجرا تست های آماری و مقاسیه عملکرد مدل های مختلف برای بحث model selection استفاده کرد.

:MCC

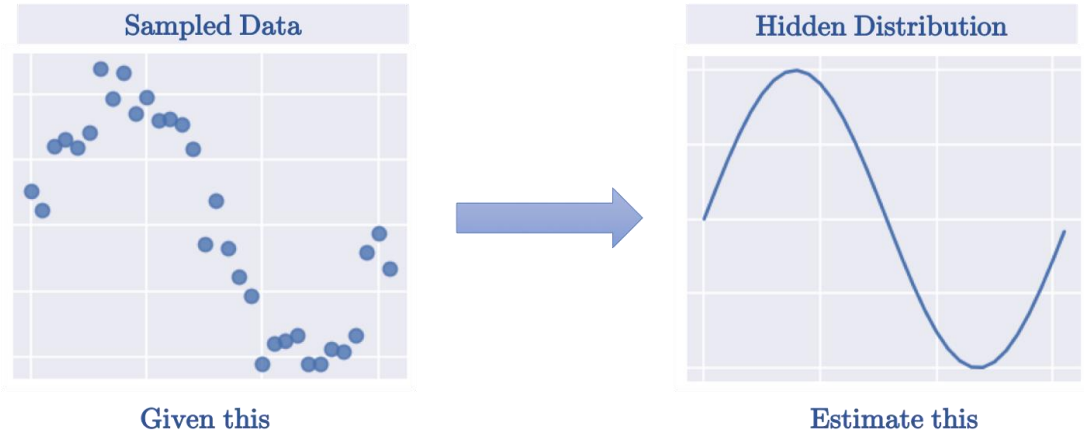
یک متریک برای محاسبه عملکرد مدل باینری یا مولتی کلاس کلاسیفایر ها می باشد که کورلیشن بین لیبل های واقعی و لیبل های پیش بینی شده را به عنوان یک آماره به نام phi coefficient به ما می دهد که عددی بین -۱ و ۱ است که ۱ به معنی این است مدل به خوبی عمل میکند و -۱ به این معنی است که تمام پیش بینی ها باید برعکس شوند. مقدار صفر نیز یعنی مدل با پرتاب سکه فرقی ندارد. این مقدار کورلیشن برای باینری کلاسیفایر به شکل زیر محاسبه میشود.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

۱۰. هدف مدل های ماشین لرنینگ به طور کلی پیدا کردن توزیع پنهان دیتا ها در یک دیتاست اصلی است. حال اگر این دیتاست اصلی بدون نویز باشد یا نویز کمتری داشته باشد توزیع که مدل ما بدست بیارد دقیق تر است و خطای کمتری دارد.



اما مشکلی که وجود دارد این است که در دنیای واقعی دیتاهایی که ما بدست می آوریم مقدار زیادی داده ی نویز دارند که باعث افزایش خطا در مدل پیش بینی شده ی ما می کند.



نمودار داده شده در سند تمرین نشان دهنده ی Bias-variance tradeoff می باشد. قبل از اینکه به تحلیل نمودار بپردازیم این دو ترم را توضیح می دهیم

Bias is termed as the error between the average model prediction f`(x)and the ground truth f(x)

$$bias = \mathbb{E}[f'(x)] - f(x)$$

$$\mathbb{E}[f'(x)] = \frac{1}{n}\sum_{i=0}^{n-1} f'_i(x)$$

مقدار دقت و خطا مقادیر پیش بینی شده توسط مدل ما به میزان پیچیدگی و درجه مدل بستگی دارد هر چقدر این مقدار کمتر باشد مقدار خطا و بایاس ما افزایش می یابند چون مقادیر پیش بینی شده با مقادیر واقعی تفاوت زیادی دارند. (underfitting)

Variance refers to the average variability in the model prediction for the given dataset.

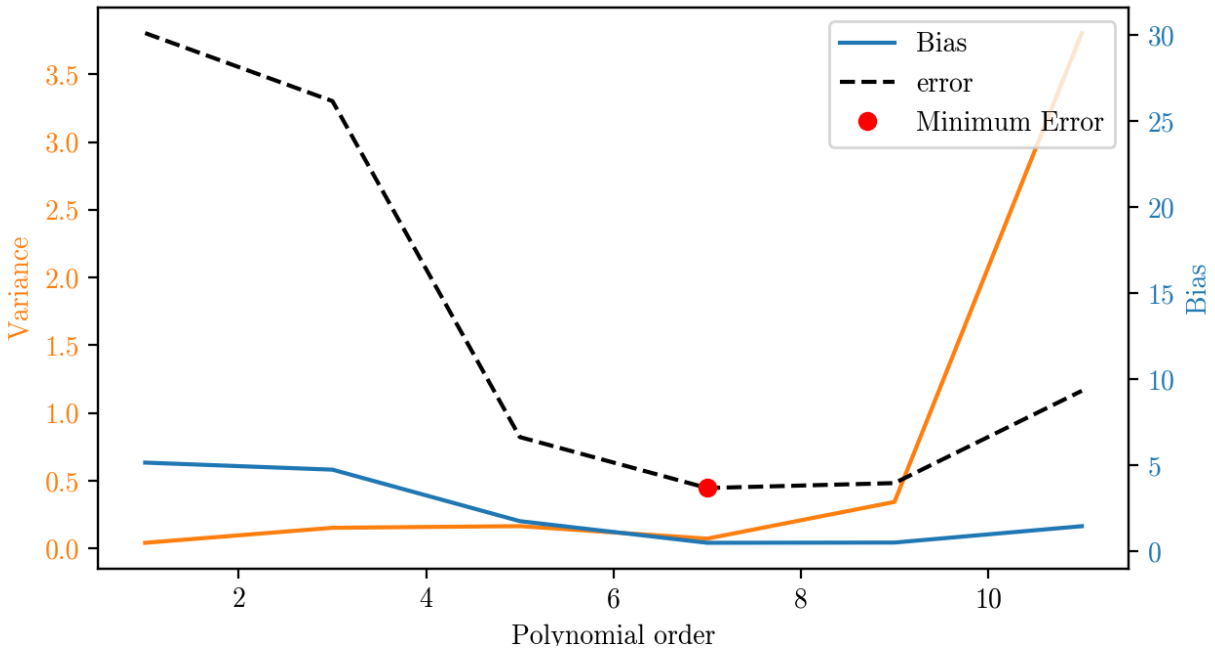
$$variance = \mathbb{E}\left[\left(f'(x) - \mathbb{E}[f'(x)]\right)^2\right]$$

حال هر چقدر مقدار پیچیدگی و درجه مدل بیشتر باشد مقدار واریانس افزایش پیدا میکند برای اینکه مدل سعی می کند به صورت کامل رفتار دیتای واقعی و حتی رفتار دیتا های اوتلایر و نویز را پیش بینی کند و باعث می شود دیگر مدل با دیتا سمپل دیگری با همان دیستریبوشن سازگار نباشد و خطای تست را بالا ببرد. (overfitting)

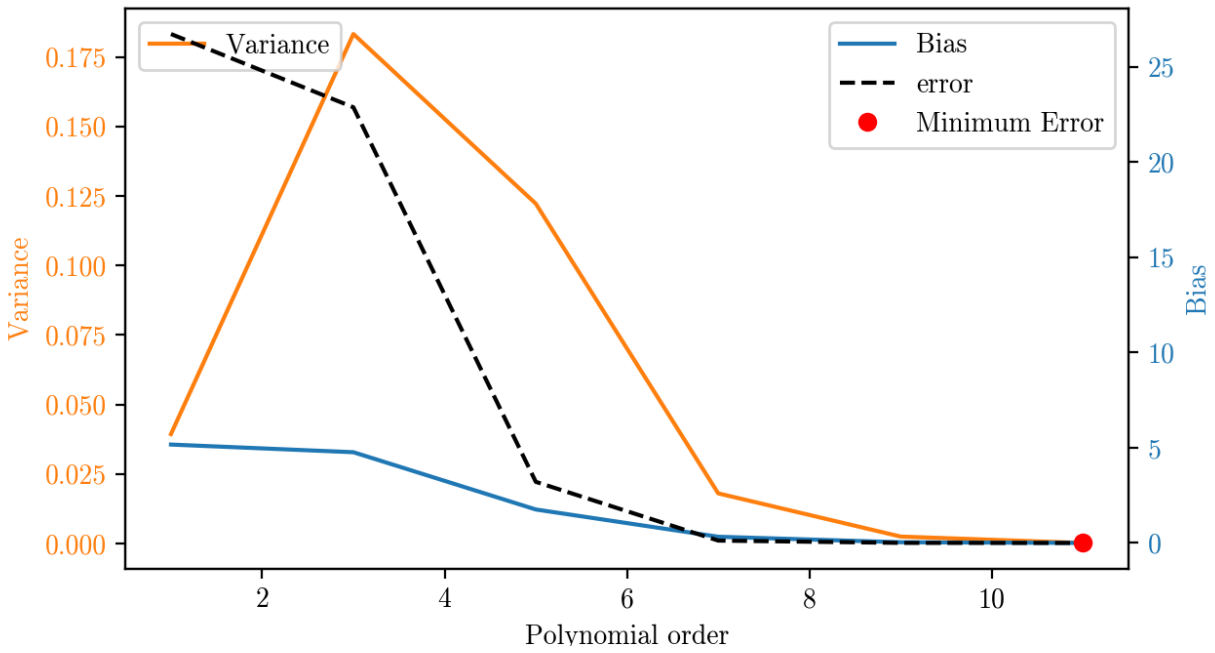
حال نمودار داده شده در سند تمرین توسط فرمول زیر کشیده شده که توضیح آن در اینجا آمده است.

$$\text{test error} = \text{bias}^2 + \text{variance} + \text{irreducible error}$$

که irreducible error همان خطایی است که توسط نویز داخل دیتاست اصلی بوجود آمده است و قابل کاهش دادن نیست تا به مدل مطلوبی برسیم.



همانطور که در شکل بالا مشاهده میشود معمولا توسط دیتاست های واقعی با افزایش پیچیدگی مدل مقدار بایاس کاهش و مقدار واریانس افزایش پیدا میکند و در این گونه مسایل ما سعی می کنیم تا مقدار پیچیدگی و درجه را پیدا کنیم که مجموع واریانس و بایاس کمینه شود. که این مقدار بسته مقدار سایز دیتاست ترینینگ و نویز داخل اصلی بستگی دارد و همیشه نمیتوان از elbow متد استفاده کرد برای این که فرض کنید دیتاستی کاملا بدون نویز یا مقدار نویز بسیار کمی داشته باشد اگر همان نمودار را برای این دیتاست بکشیم خواهیم داشت.



که همانطور که مشاهده میشود هم واریانس و هم بایاس برای این دیتا برای مدل هایی با پیچیدگی بیشتر کمتر است بنابراین خطای کلی مدل نیز کاهش یافته است بنابراین به جای استفاده از elbow متد میتوان این نمودار را برای دیتا های مختلف رسم کرد و مقدار پیچیدگی مدل را بر اساس این نمودار بدست آورد. (طوری که مقدار خطا کمینه شود)

Statistical Significance Tests for Comparing Machine Learning Algorithms:

برای مقایسه مدل های ماشین لرنینگ چندین راه متفاوت وجود دارد که به بررسی چند تا از آن ها می پردازیم. اولین روشی که وجود دارد و بسیار شایع است این است که با استفاده از k-fold cross validation نتایج عملکرد مدل را بر روی دیتای تولید شده بدست آوریم و میانگین بگیریم و مدلی که نتایج بهتری داشت را انتخاب کنیم. مشکلی که این روش دارد این است که ما صرفا با انجام این کار مطمین نخواهیم بود که اختلافات داخل نتایج واقعی بودند و یا یک اتفاق آماری رخ داده که به همچین نتیجه رسیدیم. کار دیگری که میتوان انجام داد این است که یک آزمون فرض بر روی نتایج بدست آمده اجرا کنیم تا مطمین شویم نتایجی که بدست آوردیم درست هستند. ولی مشکلی هست این است که در هنگام انجام cross validation هر رکورد به ازای k-۱ بار داخل دیتای ترین قرار می گیرد و نتیجه ای که از این میگیریم این است که نتایج ما به نحوی به یکدیگر وابسته هستند و مثلا اگر بر روی این دیتا t-test اجرا کنیم نتایج اشتباهی به ما می دهند. حال بعضی از روش هایی که به این مشکلات بر نمی خوردند در پایین لیست شده اند.

McNemar’s test : که مانند chi۲ test می باشد به این صورت تفاوت قسمت های مشاهده شده در contingency table را بررسی میکند که این تفاوت مقدار قابل توجهی هست یا خیر که معمولا برای مدل های بزرگ و عمیق شبکه عصبی کاربرد دارد.

cross validation ۲*۵: که در بالاتر توضیح داده شده، نتایج مدل ها بر روی split های داده شده توسط این روش توسط t-test بررسی میشوند. نتایج نشان داده اند که این روش محدودیت درجه آزادی بین وابستگی های نتایج داده شده را بیشتر نشان می دهد. این روش مشتقات دیگری نیز دارد مثلا نشان داده شده است ۱۰۰ بار cross validation ۱۰*۱۰ را اجرا کرد و بررسی ها را مانند قبل انجام داد و از جواب بدست آمده اطمینان بیشتری داشت.