به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس داده کاوی

پروژه نهایی

محمد ناصری

۸۱۰۱۰۰۴۸۶

خرداد ماه ۱۴۰۰

مقدمه

در این پروژه قصد داریم به تبدیل مساله‌ی تخمین خسارت وارد شده به ساختمانها در اثر زلزله براساس ویژگیهای آنها در قالب یک مساله‌ی طبقه‌بندی بپردازیم. از جمله مواردی که میتوانند بر روی میزان خسارتی که به یک ساختمان در زلزله وارد میشود تاثیر بگذارند میتوان به منطقه‌ی جغرافیایی که ساختمان در آن قرار دارد، تعداد طبقات ساختمان، مصالح مورد استفاده در ساختمان، هدف ساخت ساختمان، تعدا افراد و خانواده‌هایی که در ساختمان زندگی میکنند و وضعیت مالکیت قانونی زمین اشاره کرد. در این پروژه شما با مشکلاتی که ممکن است در طی فرآیند طبقه‌بندی ایجاد شود (مانند زیاد بودن تعداد ویژگیها، مشکل Label ،Imbalanceو غیره) مواجه شده و برای حل آنها اقدام میکنید. مجموعه داده‌ی این پروژه اطلاعات ساختمانهای آسیب دیده در زلزله‌ی Gorkhaجمع‌آوری شده ست. این پروژه شامل سه مرحله است که مرحله‌ی پایانی امتیازی است. در مرحله‌ی اول، شما باید یک روش مناسب برای آماده‌سازی ویژگیها ارائه دهید، در مرحله‌ی دوم یک مدل مناسب برای انجام هر کدام از مسائل بیابید. نهایتا در مرحله‌ی سوم شما با استفاده از مهندسی ویژگیها، یک مدل تحلیل پذیر با عملکرد بالا بدست خواهید آورد.

شرح پروژه

یکی از مشکلاتی که در مواجهه با بلایای طبیعی مانند زلزله با آن مواجه هستیم تخمین میزان خسارتی است که این زلزله به بار می‌آورد. با داشتن این اطلاعات میتوان از خسارت‌های وحشتناک بعدی با افزایش مقاومت ساختمانها یا پیش‌بینی‌های درست جلوگیری کرد. در این پروژه مجموعه اطلاعات مرتبط با ساختمانها و میزان خسارت آنها در زلزله‌ی Gorkhaدر اختیار شما قرار گرفته است. میزان خسارت وارد شده به ساختمانها به صورت یک عدد بین یک تا سه داده شده است که عدد ۱بیانگر خسارت کم، عدد ۲بیانگر خسارت متوسط و عدد ۳بیانگر تخریب کامل ساختمان است. در این پروژه ما به پیش‌بینی میزان خسارت وارد شده به ساختمانها در زمان زلزله میپردازیم که میتوان مساله‌ی تخمین آن را به صورت یک مساله ی طبقه‌بندی چندکلاسه مدل کرد.

مجموعه داده شامل اطلاعات مربوط به ساختار ساختمانها و مالکیت قانونی آنهاست. هر ردیف در مجموعه داده نشان دهنده‌ی یک ساختمان خاص در منطقه است که در زلزله‌ی Gorkhaآسیب دیده است. ۳۹ویژگی ۲در این مجموعه وجود دارد که ستون Bulding_idیک شناسه‌ی یکتا و تصادفی برای هر ساختمان است .۳۸ویژگی باقیمانده را در لیست زیر مشاهده میکنید

- *Description of features*
  - ***geo_level_1_id, geo_level_2_id, geo_level_3_id (type: int):*** geographic region in which building exists, from largest (level 1) to most specific sub-region (level 3). Possible values: level 1: 0-30, level 2: 0-1427, level 3: 0-12567.
  - ***count_floors_pre_eq (type: int):*** number of floors in the building before the earthquake.
  - ***age (type: int):*** age of the building in years.
  - ***area_percentage (type: int):*** normalized area of the building footprint.
  - ***height_percentage (type: int):*** normalized height of the building footprint.
  - ***land_surface_condition (type: categorical):*** surface condition of the land where the building was built. Possible values: n, o, t.
  - ***foundation_type (type: categorical):*** type of foundation used while building. Possible values: h, i, r, u, w.
  - ***roof_type (type: categorical):*** type of roof used while building. Possible values: n, q, x.
  - ***ground_floor_type (type: categorical):*** type of the ground floor. Possible values: f, m, v, x, z.
  - ***other_floor_type (type: categorical):*** type of constructions used in higher than the ground floors (except of roof). Possible values: j, q, s, x.
  - ***position (type: categorical):*** position of the building. Possible values: j, o, s, t.
  - ***plan_configuration (type: categorical):*** building plan configuration. Possible values: a, c, d, f, m, n, o, q, s, u.
  - ***has_superstructure_adobe_mud (type: binary):*** flag variable that indicates if the superstructure was made of Adobe/Mud.
  - ***has_superstructure_mud_mortar_stone (type: binary):*** flag variable that indicates if the superstructure was made of Mud Mortar - Stone.
  - ***has_superstructure_stone_flag (type: binary):*** flag variable that indicates if the superstructure was made of Stone.
  - ***has_superstructure_cement_mortar_stone (type: binary):*** flag variable that indicates if the superstructure was made of Cement Mortar - Stone.
  - ***has_superstructure_mud_mortar_brick (type: binary):*** flag variable that indicates if the superstructure was made of Mud Mortar - Brick.
  - ***has_superstructure_cement_mortar_brick (type: binary):*** flag variable that indicates if the superstructure was made of Cement Mortar - Brick.
  - ***has_superstructure_timber (type: binary):*** flag variable that indicates if the superstructure was made of Timber.
  - ***has_superstructure_bamboo (type: binary):*** flag variable that indicates if the superstructure was made of Bamboo.
  - ***has_superstructure_rc_non_engineered (type: binary):*** flag variable that indicates if the superstructure was made of non-engineered reinforced concrete.
  - ***has_superstructure_rc_engineered (type: binary):*** flag variable that indicates if the superstructure was made of engineered reinforced concrete.
  - ***has_superstructure_other (type: binary):*** flag variable that indicates if the superstructure was made of any other material.
  - ***legal_ownership_status (type: categorical):*** legal ownership status of the land where building was built. Possible values: a, r, v, w.
  - ***count_families (type: int):*** number of families that live in the building.
  - ***has_secondary_use (type: binary):*** flag variable that indicates if the building was used for any secondary purpose.
  - ***has_secondary_use_agriculture (type: binary):*** flag variable that indicates if the building was used for agricultural purposes.
  - ***has_secondary_use_hotel (type: binary):*** flag variable that indicates if the building was used as a hotel.
  - ***has_secondary_use_rental (type: binary):*** flag variable that indicates if the building was used for rental purposes.
  - ***has_secondary_use_institution (type: binary):*** flag variable that indicates if the building was used as a location of any institution.
  - ***has_secondary_use_school (type: binary):*** flag variable that indicates if the building was used as a school.
  - ***has_secondary_use_industry (type: binary):*** flag variable that indicates if the building was used for industrial purposes.
  - ***has_secondary_use_health_post (type: binary):*** flag variable that indicates if the building was used as a health post.
  - ***has_secondary_use_gov_office (type: binary):*** flag variable that indicates if the building was used fas a government office.
  - ***has_secondary_use_use_police (type: binary):*** flag variable that indicates if the building was used as a police station.
  - ***has_secondary_use_other (type: binary):*** flag variable that indicates if the building was secondarily used for other purposes.

مرحله اول: آماده‌سازی ویژگی‌ها

در ابتدا باید داده‌ها را برای استفاده در یک مدل یادگیری ماشین آماده کنیم. با توجه به اینکه ویژگی‌های داده شده شامل انواع مختلفی از متغیرها هستند، نیاز است برای تبدیل آنها به متغیرهای عددی تمهیداتی اندیشیده شود، زیرا اکثریت مدل‌های مبتنی‌بر یادگیری ماشین تنها بر روی داده‌های عددی کار میکنند. در این مرحله ما باید ویژگیهای داده شده را برای استفاده در مدل‌های یادگیری ماشین آماده کنیم. به علاوه، با توجه به تعداد زیاد ویژگیها، ممکن است استفاده از تمامی آنها در یک مدل یادگیری ماشین به بیش‌برازش منجر بشود. بنابراین، یکی دیگر از وظایف ما در این بخش کاهش ابعاد ویژگی‌ها میباشد.

بررسی اولیه مجموعه داده

در این مجموعه داده ما ۲۶۰۶۰۱ سطر(ساختمان) متفاوت را مورد بررسی قرار میدهیم. به ازای هرکدام از این سطرها تعداد ۳۹ فیچر داریم که فیچرها و نوع داده آنها و تعداد مقادیر NULL و مقادیر یکتا برای هر ویژگی در جدول زیر قابل مشاهده است.

	Data Type	Unique Values	Null Values	% null Values
building_id	int64	260601	0	0.0
geo_level_1_id	int64	31	0	0.0
geo_level_2_id	int64	1414	0	0.0
geo_level_3_id	int64	11595	0	0.0
count_floors_pre_eq	int64	9	0	0.0
age	int64	42	0	0.0
area_percentage	int64	84	0	0.0
height_percentage	int64	27	0	0.0
land_surface_condition	object	3	0	0.0
foundation_type	object	5	0	0.0
roof_type	object	3	0	0.0
ground_floor_type	object	5	0	0.0
other_floor_type	object	4	0	0.0
position	object	4	0	0.0

plan_configuration	object	10	0	0.0
has_superstructure_adobe_mud	int64	2	0	0.0
has_superstructure_mud_mortar_stone	int64	2	0	0.0
has_superstructure_stone_flag	int64	2	0	0.0
has_superstructure_cement_mortar_stone	int64	2	0	0.0
has_superstructure_mud_mortar_brick	int64	2	0	0.0
has_superstructure_cement_mortar_brick	int64	2	0	0.0
has_superstructure_timber	int64	2	0	0.0
has_superstructure_bamboo	int64	2	0	0.0
has_superstructure_rc_non_engineered	int64	2	0	0.0
has_superstructure_rc_engineered	int64	2	0	0.0
has_superstructure_other	int64	2	0	0.0
legal_ownership_status	object	4	0	0.0
count_families	int64	10	0	0.0
has_secondary_use	int64	2	0	0.0
has_secondary_use_agriculture	int64	2	0	0.0
has_secondary_use_hotel	int64	2	0	0.0
has_secondary_use_rental	int64	2	0	0.0
has_secondary_use_institution	int64	2	0	0.0
has_secondary_use_school	int64	2	0	0.0
has_secondary_use_industry	int64	2	0	0.0
has_secondary_use_health_post	int64	2	0	0.0
has_secondary_use_gov_office	int64	2	0	0.0
has_secondary_use_use_police	int64	2	0	0.0
has_secondary_use_other	int64	2	0	0.0

اولین مشکل چشمگیر در بررسی این مجموعه داده تعداد بالای ویژگی‌ها و سطرهای آن است که باعث ایجاد پدیده Curse of dimensionality میشوند. لازم است تا با روش هایی از تعداد ویژگی‌ها بکاهیم.

پس از بررسی نوع داده‌های ویژگی‌های جدول باید ببینیم این ویژگی‌ها چه مقادیری را میگیرند. برای این کار ابتدا در جدول زیر مقادیر عددی را مقایسه میکنیم.

	count	mean	std	min	25%	50%	75%	max
building_id	260601.000000	525675.482773	304544.999032	4.000000	261190.000000	525757.000000	789762.000000	1052934.000000
geo_level_1_id	260601.000000	13.900353	8.033617	0.000000	7.000000	12.000000	21.000000	30.000000
geo_level_2_id	260601.000000	701.074685	412.710734	0.000000	350.000000	702.000000	1050.000000	1427.000000
geo_level_3_id	260601.000000	6257.876148	3646.369645	0.000000	3073.000000	6270.000000	9412.000000	12567.000000
count_floors_pre_eq	260601.000000	2.129723	0.727665	1.000000	2.000000	2.000000	2.000000	9.000000
age	260601.000000	26.535029	73.565937	0.000000	10.000000	15.000000	30.000000	995.000000
area_percentage	260601.000000	8.018051	4.392231	1.000000	5.000000	7.000000	9.000000	100.000000
height_percentage	260601.000000	5.434365	1.918418	2.000000	4.000000	5.000000	6.000000	32.000000
has_superstructure_adobe_mud	260601.000000	0.088645	0.284231	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_mud_mortar_stone	260601.000000	0.761935	0.425900	0.000000	1.000000	1.000000	1.000000	1.000000
has_superstructure_stone_flag	260601.000000	0.034332	0.182081	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_cement_mortar_stone	260601.000000	0.018235	0.133800	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_mud_mortar_brick	260601.000000	0.068154	0.252010	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_cement_mortar_brick	260601.000000	0.075268	0.263824	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_timber	260601.000000	0.254988	0.435855	0.000000	0.000000	0.000000	1.000000	1.000000

has_superstructure_bamboo	260601.000000	0.085011	0.278899	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_rc_non_engineered	260601.000000	0.042590	0.201931	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_rc_engineered	260601.000000	0.015859	0.124932	0.000000	0.000000	0.000000	0.000000	1.000000
has_superstructure_other	260601.000000	0.014985	0.121491	0.000000	0.000000	0.000000	0.000000	1.000000
count_families	260601.000000	0.983949	0.418389	0.000000	1.000000	1.000000	1.000000	9.000000
has_secondary_use	260601.000000	0.111880	0.315219	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_agriculture	260601.000000	0.064378	0.245426	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_hotel	260601.000000	0.033626	0.180265	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_rental	260601.000000	0.008101	0.089638	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_institution	260601.000000	0.000940	0.030647	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_school	260601.000000	0.000361	0.018989	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_industry	260601.000000	0.001071	0.032703	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_health_post	260601.000000	0.000188	0.013711	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_gov_office	260601.000000	0.000146	0.012075	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_use_police	260601.000000	0.000088	0.009394	0.000000	0.000000	0.000000	0.000000	1.000000
has_secondary_use_other	260601.000000	0.005119	0.071364	0.000000	0.000000	0.000000	0.000000	1.000000

همانطور که مشاهده میشود داده‌های عددی این مجموعه ویژگی‌ها، شامل تعدادی داده باینری، تعدادی داده در بازه متفاوت و تعدادی داده که نشانگر شناسه (id) هستند میباشد. مشکلی که در این داده‌ها مشاهده میشود اختلاف بازه اعداد برای داده‌های باینری و دیگر داده‌ها به‌خصوص داده‌های شناسه است. این اتفاق باعث میشود تا اعداد بزرگتر تاثیر بیشتری در طقبه بندی داشته

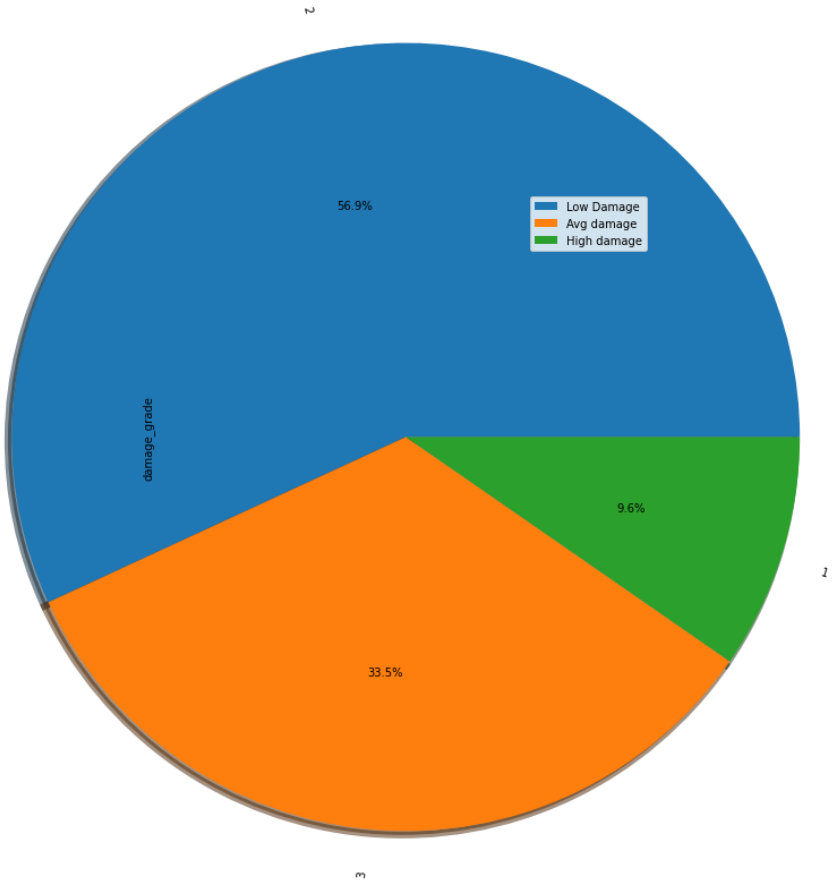
باشند و باعث ایجاد خطا و طبقه‌بندی اشتباه بشوند. برای حل این مشکل راهکار اولیه نرمال کردن داده‌هاست. مساله ما اینجا این است که چگونه داده‌های عددی داده شده را نرمالسازی کنیم.

داده‌های Categorical نیز در جدول زیر قابل مشاهده هستند:

	count	unique	top	freq
land_surface_condition	260601	3	t	216757
foundation_type	260601	5	r	219196
roof_type	260601	3	n	182842
ground_floor_type	260601	5	f	209619
other_floor_type	260601	4	q	165282
position	260601	4	s	202090
plan_configuration	260601	10	d	250072
legal_ownership_status	260601	4	v	250939

مشکل در مواجهه با داده‌های Categorical این است که اغلب الگوریتم‌های یادگیری ماشین تنها قادر به پردازش اعداد هستند و در نتیجه باید با روشی این داده‌ها را به مقادیر عددی تبدیل کرد که در ادامه به آن میپردازیم.

از موارد تاثیرگذار دیگر در دقت یادگیری ماشین توزیع کلاس‌های مختلف در داده است.

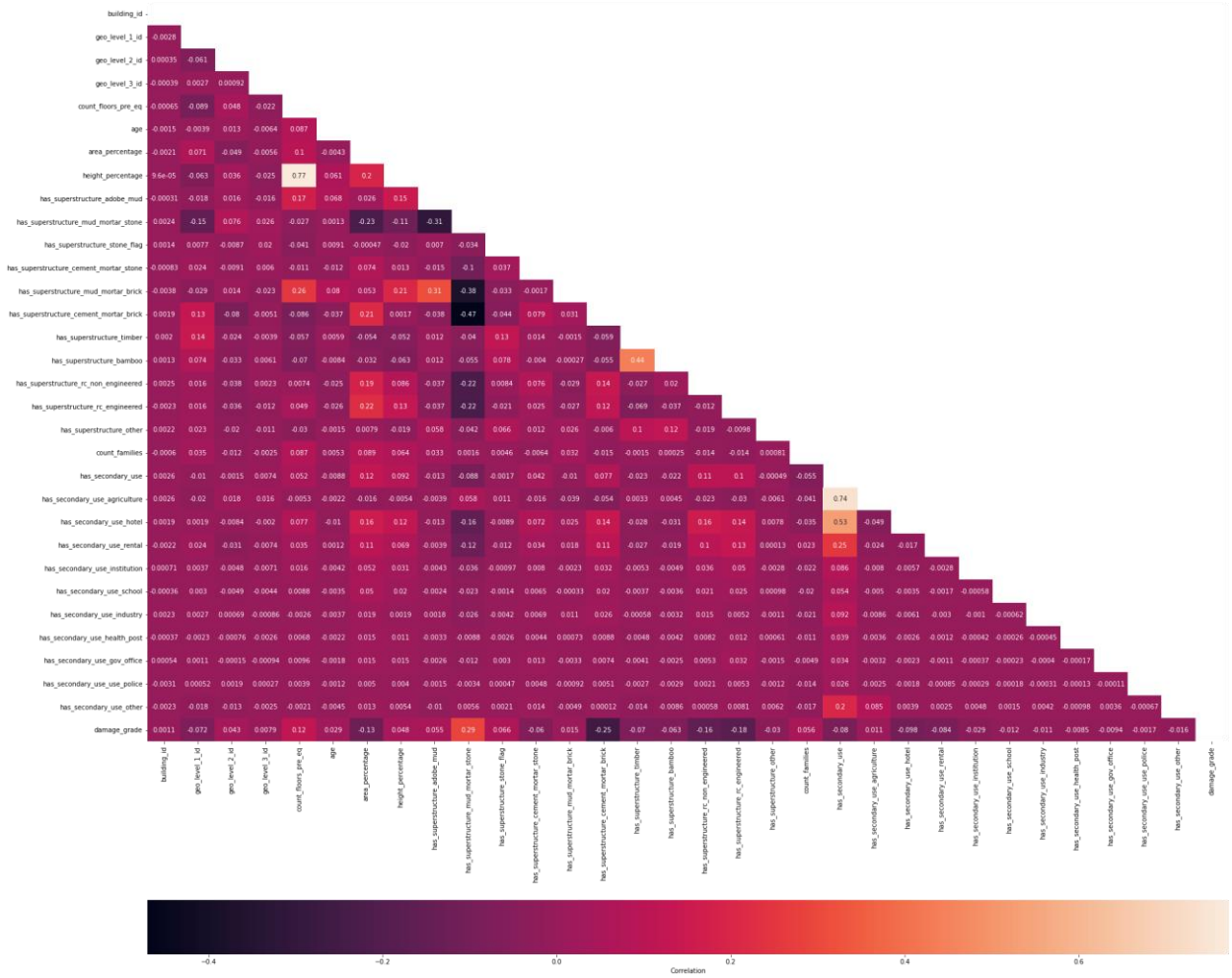


مشکلی که در این قسمت مشاهده میشود این است که کلاس‌ها توزیع یکسانی ندارند و برای مثال از گروه ۱ داده بسیار کمتری نسبت به گروه ۲ یا ۳ در اختیار داریم. یک مسئله طبقه بندی نامتعادل نمونه ای از یک مشکل طبقه بندی است که در آن توزیع مثال ها در بین کلاس های شناخته شده مغرضانه یا کج است. توزیع داده‌ها می تواند از یک سوگیری جزئی تا یک عدم تعادل شدید متفاوت باشد. طبقه‌بندی‌های نامتعادل چالشی برای مدل‌سازی ایجاد می‌کنند، زیرا بسیاری از الگوریتم‌های یادگیری ماشینی که برای طبقه‌بندی استفاده می‌شوند، بر اساس فرض تعداد مساوی نمونه برای هر کلاس طراحی شده‌اند. این منجر به مدل هایی می شود که عملکرد پیش بینی ضعیفی دارند، به ویژه برای کلاس اقلیت، که یک مشکل است زیرا معمولاً کلاس اقلیت مهم‌تر است و بنابراین مشکل به اشتباهات طبقه‌بندی برای کلاس اقلیت نسبت به کلاس اکثریت حساس‌تر است.

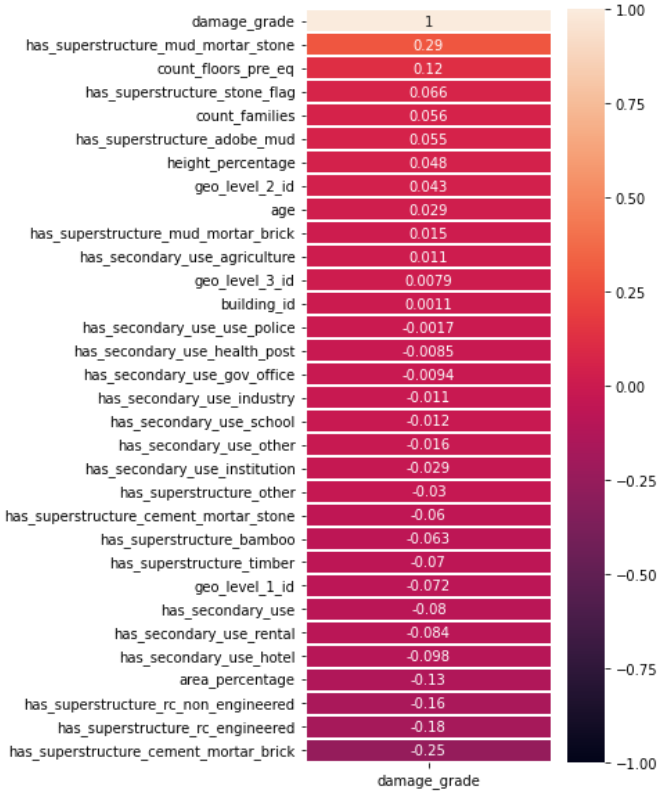
اقدام برای رفع مشکلات مجموعه داده

1 - تعداد ویژگی

اولین مشکل مشاهده شده در دیتاست، تعداد زیاد ویژگی‌های آن است. برای رفع این مشکل نیاز است تا تعدادی از آنها را حذف و یا به نحوی کاهش دهیم. برای حذف ویژگی‌ها یکی از ساده‌ترین راهها بررسی وجود همبستگی بالا میان یک ویژگی با ویژگی دیگر است. در این صورت به علت این همبستگی بالا این ۲ ویژگی تاثیر تقریبا یکسانی خواهند داشت و میتوان یکی از این ۲ را حذف کرد. برای این کار از ماتریس correlation ویژگی ها استفاده میکنیم.



همانطور که مشاهده میشود در این ماتریس تعدادی از ویژگی ها با یکدیگر همبستگی بالایی دارند که میتوانیم یکی از آنها را به قرینه دیگری حذف بنماییم.



راه ساده دیگر برای حذف ویژگی، بررسی تاثیر آنها در نتیجه نهایی است. برای این کار از جدول همبستگی زیر استفاده میکنیم. (لازم به ذکر است در این جدول و جدول قبلی از ذات ترتیبی بودن لیبل کلاس‌ها یعنی میزان تخریب و صدمه برای بدست آوردن همبستگی‌ها استفاده شده است.)

همانطور که در جدول قابل مشاهده است تعدادی از ویژگی‌ها تاثیر بسیار کمی در نتیجه نهایی تخریب داشته و میتوان از آنها چشم پوشی کرد.

همچنین روش‌های کاهش ویژگی دیگری مانند PCAو LDA میتوان استفاده کرد. (این موارد پس از تست، بهبود نتیجه قابل توجهی نداشتند فلذا از استفاده کردن از آنها صرف نظر شد)

۲- ویژگی‌های Categorical

برخی از الگوریتم‌ها می‌توانند مستقیماً با داده‌های طبقه‌بندی کار کنند .به عنوان مثال، یک درخت تصمیم می تواند مستقیماً از داده های طبقه بندی بدون نیاز به تبدیل داده یادگیری را انجام دهد. بسیاری از الگوریتم های یادگیری ماشینی نمی توانند مستقیماً روی داده های برچسب کار کنند و نیاز دارند که همه متغیرهای ورودی و متغیرهای خروجی عددی باشند. به طور کلی، این یک محدودیت برای اجرای کارآمد الگوریتم‌های یادگیری ماشین است. این بدان معناست که داده های دسته بندی باید به فرم عددی تبدیل شوند. برای اینکار از روش‌های متفاوتی میتوان استفاده کرد. ساده‌ترین روش جایگذاری هر برچسب متعلق به دسته خاص با یک عدد است. مشکل اصلی این روش این است که اکثر داده‌های موجود ما ذات ترتیبی ندارند ولی در این روش برای داده‌ها ترتیب در نظر گرفته میشود یعنی برای مثال اگر در دسته بندی Land_surface_condition به جای ۳ برچسب (n,o,t) از (۱و۲و۳) استفاده کنیم در واقع برای n مقدار بالاتری نسبت به t در نظر گرفته‌ایم در حالیکه در واقعیت ترتیبی میان آنها وجود ندارد.

در نتیجه برای متغیرهای طبقه ای که رابطه ترتیبی وجود ندارد، رمزگذاری عددی صحیح کافی نیست. در واقع، استفاده از این رمزگذاری و اجازه دادن به مدل برای در نظر گرفتن نظم طبیعی بین دسته‌ها ممکن است منجر به عملکرد ضعیف یا نتایج غیرمنتظره شود. در این مورد، یک رمزگذاری one hot می تواند برای نمایش عدد صحیح به جای برچسب‌ها اعمال شود. در اینجاست که متغیر categorical حذف می شود و یک متغیر باینری جدید برای هر برچسب منحصر به فرد اضافه می شود.

۳- نرمالسازی مقادیر

مقیاس بندی ویژگی ها در یادگیری ماشین یکی از حیاتی ترین مراحل در طول پیش پردازش داده ها قبل از ایجاد یک مدل یادگیری ماشین است. مقیاس بندی می تواند بین یک مدل یادگیری ماشین ضعیف و یک مدل بهتر تفاوت ایجاد کند.

متداول ترین تکنیک های مقیاس بندی ویژگی ها نرمالسازی و استانداردسازی هستند.

نرمال سازی زمانی استفاده می شود که بخواهیم مقادیر خود را بین دو عدد، معمولاً بین [0،1] یا [-1،1] محدود کنیم. در حالی که استانداردسازی داده ها را به میانگین صفر و واریانس 1 تبدیل می کند و داده های ما را بدون واحد می کنند.

الگوریتم‌های یادگیری ماشین فقط اعداد را می‌بینند - اگر تفاوت زیادی در بازه اعداد وجود داشته باشد، این فرض را ایجاد می‌کند که اعداد با محدوده بالاتر به نوعی برتری دارند. بنابراین، این تعداد ویژگی‌ها، در حین آموزش مدل، نقش تعیین کننده تری ایفا می کنند. الگوریتم‌های یادگیری ماشین روی اعداد کار می کند و نمی دانند که آن عدد چه چیزی را نشان می دهد. وزن 10 گرم و قیمت 10 دلار نشان‌دهنده دو چیز کاملاً متفاوت است - که برای انسان مشخص است، اما برای یک مدل به عنوان یک ویژگی، هر دو را یکسان می‌داند.

روش‌های متفاوتی برای این کار وجود دارند که برخی آنها عبارتند از:

1) Min Max Scaler
2) Standard Scaler
3) Max Abs Scaler
4) Robust Scaler
5) Quantile Transformer Scaler
6) Power Transformer Scaler
7) Unit Vector Scaler

در این مساله ما با ۲ دسته از داده‌های عددی روبرو هستیم. (صرف نظر از باینری) دسته اول مربوط به اعداد صحیح(integer) مانند سن و ... هستند و دسته دوم شامل شناسه‌های عددی میباشند. شناسه‌های عددی در واقع نوعی از داده Categorical هستند ولی به علت عددی بودن در این قسمت آورده شده‌اند. برای رفع مشکل بازه اعداد صحیح از standard_scaler استفاده میکنیم. standard_scaler فرض می‌کند که داده‌ها به طور نرمال در هر ویژگی توزیع می‌شوند و آن‌ها را به‌گونه‌ای مقیاس‌بندی می‌کند که توزیع حول محور ۰، با انحراف استاندارد ۱ باشد.

در مقابل برای شناسه‌های عددی (داده‌های geo) از target encoding استفاده میکنیم. ایده اصلی پشت target encoding این است که دسته‌ها را با جایگزین کردن آنها با اندازه‌گیری تأثیری که ممکن است روی هدف بگذارند، رمزگذاری کند. در یک طبقه‌بندی‌کننده باینری، ساده‌ترین راه برای انجام این کار، محاسبه احتمال $p(t = 1 \mid x = ci)$ است که در آن t نشان‌دهنده هدف، x ورودی و ci رده i-ام است. در آمار بیزی، با توجه به اینکه ورودی رده ci بوده است، این احتمال، احتمال پسین t=1 در نظر گرفته می شود. این به این معنی است که ما رده ci را برای مقدار احتمال پسین یک بودن هدف در حضور آن دسته جایگزین می کنیم.

مرحله‌ی دوم: یادگیری و انتخاب مدل مناسب

مدل‌سازی

با توجه به بررسی‌های انجام شده، اقدامات لازم را انجام داده و به ساختن مدلی اولیه و ساده برای بررسی نتایج میپردازیم. برای این کار از طبقه‌بند SVM استفاده میکنیم.

نکته: برای پیاده‌سازی‌های انجام شده با توجه به ابعاد بسیار بالای داده و در دست داشتن منابع کم برای اجرای الگوریتم‌ها بر روی تمام دیتاست، از داده‌ها نمونه گیری شده و الگوریتم‌ها روی نمونه اجرا میشوند و در نهایت بر روی دیتاست اصلی اجرا خواهند شد.

پس از اجرای اولیه الگوریتم svm رو داده خام(فقط One hot انجام شده) نتایج زیر حاصل میشود:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	16879
2	0.57	1.00	0.73	100077
3	0.00	0.00	0.00	58949
accuracy			0.57	175905
macro avg	0.19	0.33	0.24	175905
weighted avg	0.32	0.57	0.41	175905

مقادیر دقت و معیارهای دیگر برای این نتایج بسیار ضعیف برای تلاش برای بهبود نتایج، تبدیل‌های تعریف شده در قسمت قبل را اعمال میکنیم تا دیتاست جدید بدست آوریم.

همانطور که ذکر شد در این مساله با ۲ طبقه‌بند سر و کار خواهیم داشت: ۱- SVM و ۲-MLP

Support Vector Machin (SVM)

ماشین بردار پشتیبان (SVM) یک مدل یادگیری ماشینی نظارت شده است که از الگوریتم‌های طبقه‌بندی برای مسائل طبقه‌بندی دو گروهی استفاده می‌کند. در مقایسه با الگوریتم های جدیدتر مانند شبکه های عصبی، این روش دو مزیت اصلی دارد: سرعت بالاتر و عملکرد بهتر با تعداد محدود نمونه (در هزاران). این باعث می شود که الگوریتم برای مسائل طبقه بندی متن بسیار مناسب باشد، جایی که دسترسی به مجموعه داده ای از حداکثر چند هزار نمونه برچسب گذاری شده معمول است.

بردارهای پشتیبان در واقع مختصات یک مشاهده منفرد هستند. برای مثال (150، 45) یک بردار پشتیبان است که به یک زن اختصاص دارد. ماشین بردار پشتیبان مرزی است که دسته مردان و زنان را به بهترین وجه از یکدیگر جدا می‌کند. در این مثال، دو دسته وجود دارد و بنابراین جداسازی آن‌ها به وسیله ماشین بردار پشتیبان آسان است.

مزایا

- حاشیه جداسازی برای دسته‌های مختلف کاملاً واضح است.
- در فضاهای با ابعاد بالاتر کارایی بیشتری دارد.
- در شرایطی که تعداد ابعاد بیش از تعداد نمونه‌ها باشد نیز کار می‌کند.
- یک زیر مجموعه از نقاط تمرینی را در تابع تصمیم‌گیری استفاده می‌کند (که به آن‌ها بردارهای پشتیبان گفته می‌شود)، بنابراین در مصرف حافظه نیز به صورت بهینه عمل می‌کند.

معایب

- هنگامی که مجموعه داده‌ها بسیار بزرگ باشد، عملکرد خوبی ندارد، زیرا نیازمند زمان آموزش بسیار زیاد است.
- هنگامی که مجموعه داده (نویز) زیادی داشته باشد، عملکرد خوبی ندارد و کلاس‌های هدف دچار همپوشانی می‌شوند.
- ماشین بردار پشتیبان به طور مستقیم تخمین‌های احتمالاتی را فراهم نمی‌کند و این موارد با استفاده از یک اعتبارسنجی متقابل (Cross Validation) پرهزینه پنج‌گانه انجام می‌شوند. این امر با روش SVC موجود در کتابخانه scikit-kearn پایتون، مرتبط است.

هایپرپارامترهای این طبقه بند عبارتند از:

کرنل: این پارامتر پیش از این مورد بررسی قرار گرفت. گزینه‌های گوناگونی شامل«linear» ، «rbf»، «poly»برای کرنل وجود دارند و در حالت پیش‌فرض کرنل روی پارامتر rbf قرار دارد .و poly برای خط جداساز غیر راست مفید هستند.

گاما :(gamma) ضریب کرنل برایrbf ، polyو sigmoid است. هرچه مقدار گاما بیشتر باشد، الگوریتم تلاش می‌کند برازش را دقیقاً بر اساس مجموعه داده‌های تمرینی انجام دهد و این امر موجب تعمیم یافتن خطا و وقوع مشکل بیش برازش(Over-Fitting)می‌شود.

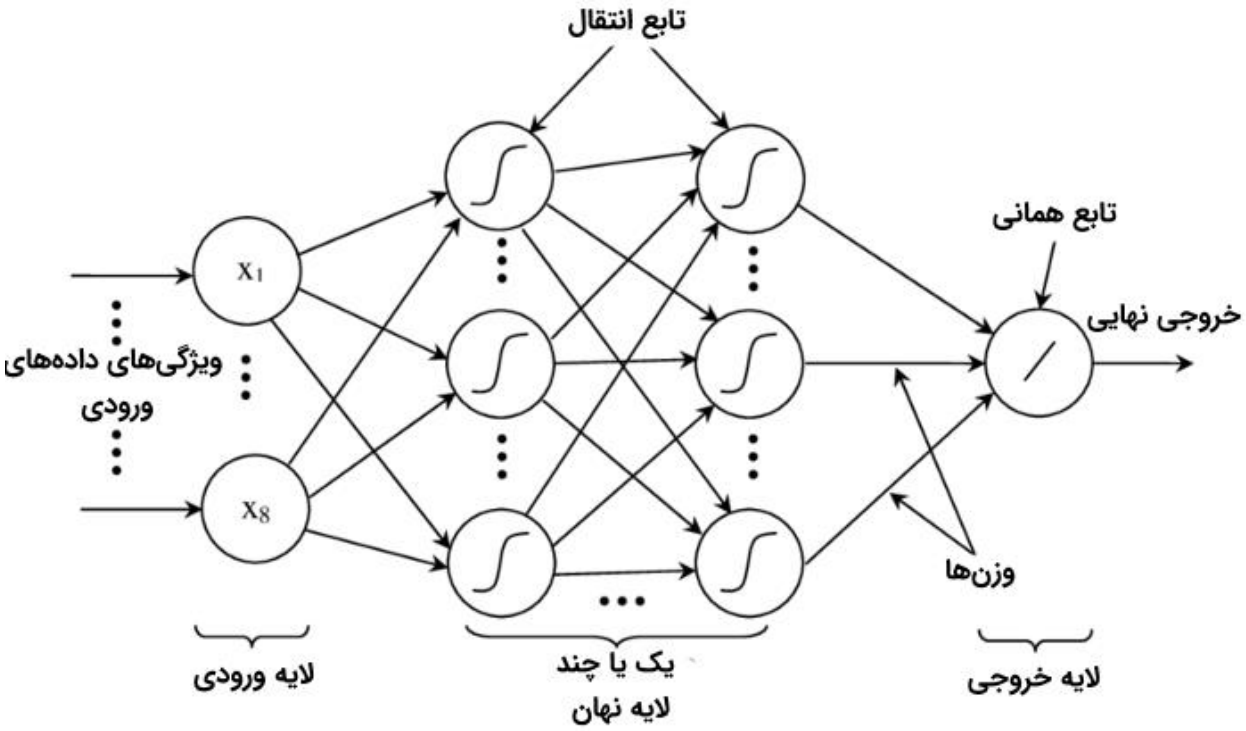
C: پارامتر جریمه C، برای جمله خطا است. این پارامتر همچنین برقراری تعادل بین مرزهای تصمیم‌گیری هموار و طبقه‌بندی نقاط داده تمرینی را کنترل می‌کند.

Multi-layer Perceptron (MLP)

شبکه عصبی پرسپترون چند لایه، دسته‌ای از شبکه‌های عصبی مصنوعی پیشخور محسوب می‌شوند. در بخش بعد، با مفهوم پیشخور در این شبکه عصبی آشنا خواهیم شد. در یک شبکه عصبی پرسپترون چند لایه، حداقل سه «لایه (Layer) «از «نودها (Nodes) « وجود خواهند داشت:

- یک «لایه ورودی(Input Layer) «
- یک «لایه نهان(Hidden Layer) «
- یک «لایه خروجی(Output layer) «

یکی از پایه‌ای‌ترین مدل‌های عصبی موجود، مدل پرسپترون چند لایه یا Multi-Layer Perceptron (به اختصارMLP) است که عملکرد انتقالی مغز انسان را شبیه‌سازی می‌کند. در این نوع شبکه عصبی، بیشتر رفتار شبکه‌ای مغز انسان و انتشار سیگنال در آن مد نظر بوده است و از این رو، گهگاه با نام شبکه‌های پیش‌خورد (Feedforward Networks) نیز خوانده می‌شوند. هر یک از سلول‌های عصبی مغز انسان، موسوم به نورون(Neuron) ، پس از دریافت ورودی (از یک سلول عصبی یا غیر عصبی دیگر)، پردازشی روی آن انجام می‌دهند و نتیجه را به یک سلول دیگر (عصبی یا غیر عصبی) انتقال می‌دهند. این رفتار تا حصول نتیجه‌ای مشخص ادامه دارد، که احتمالاً در نهایت منجر به یک تصمیم، پردازش، تفکر و یا حرکت خواهد شد.



هایپرپارامترهای مورد استفاده MLP

- **فعالساز (Activation)**: تابع فعال سازی لایه پنهان
- **حل کننده** {'lbfgs', 'sgd', 'adam'}, default='adam’

حل کننده بهینه سازی وزن

'lbfgs' یک بهینه ساز در خانواده روش های شبه نیوتنی است.

'sgd' به نزول گرادیان تصادفی اشاره دارد.

"adam" به یک بهینه ساز مبتنی بر گرادیان تصادفی اشاره دارد که توسط کینگما، دیدریک و جیمی با پیشنهاد شده است.

توجه: حل‌کننده پیش‌فرض «adam » روی مجموعه داده‌های نسبتاً بزرگ (با هزاران نمونه آموزشی یا بیشتر) از نظر زمان آموزش و امتیاز اعتبار سنجی به خوبی کار می‌کند. با این حال، برای مجموعه داده‌های کوچک، «lbfgs» می‌تواند سریع‌تر همگرا شود و عملکرد بهتری داشته باشد.

- **alpha**، پیش فرض = 0.0001 : قدرت ترم تنظیم L2.
- **Learning_rate** {'constant', 'invscaling', 'adaptive'}, default='constant’

برنامه نرخ یادگیری برای به روز رسانی وزن.

"constant" یک نرخ یادگیری ثابت است که توسط "learning_rate_init" ارائه می‌شود.

'invscaling' به تدریج نرخ یادگیری را در هر مرحله زمانی 't' با استفاده از یک توان مقیاس معکوس 'power_t' کاهش می دهد.

«تطبیقی» نرخ یادگیری را تا زمانی که از دست دادن آموزش کم می‌کند تا «نرخ_آموزش_شروع» ثابت نگه می‌دارد. هر بار که دو دوره متوالی نتوانند میزان تلفات آموزش را حداقل تا میزان TOL کاهش دهند، یا اگر «توقف زودهنگام» اتفاق بیفتد و الگوریتم نتوانند امتیاز اعتبارسنجی را حداقل تا یک میلیون افزایش دهند، نرخ یادگیری فعلی بر 5 تقسیم می‌شود.

پیدا کردن پارامترهای بهینه

برای این کار از GRIDSEARCH استفاده میکنیم. این یک راهکار تنظیم پارامتر است. نقطه کلیدی درباره عملکرد این متد این است که، برای هر ترکیب ممکن از پارامترهای الگوریتم، که در شبکه توری مشخص شده باشد، مدل را به صورت روشمند ساخته و ارزیابی می کند. از این رو ، می توان گفت که این الگوریتم ماهیت جستجو دارد.

پس از اجرا و بدست آوردن مقادیر بهینه برای دو طبقه‌بند و اجرای آنها روی داده نمونه، نتایج زیر حاصل میشود:

	precision	recall	f1-score	support
1	0.68	0.46	0.55	16879
2	0.74	0.85	0.79	100077
3	0.76	0.65	0.70	58949
accuracy			0.74	175905
macro avg	0.73	0.65	0.68	175905
weighted avg	0.74	0.74	0.74	175905

Figure 1 گزارش طبقه بندی svm

	precision	recall	f1-score	support
1	0.63	0.52	0.57	16879
2	0.76	0.81	0.78	100077
3	0.74	0.68	0.71	58949
accuracy			0.74	175905
macro avg	0.71	0.67	0.69	175905
weighted avg	0.74	0.74	0.74	175905

Figure 2گزارش طبقه بندی mlp

به طور پیش فرض، SVM معمولاً دقت پیش بینی بالاتری نسبت به پرسپترون چند لایه دارد. SVM ممکن است زمان اجرا بالاتری داشته باشد زیرا محاسباتی مانند ترجمه فضای n بعدی با استفاده از توابع کرنل پیشرفته است. اما معمولاً در پیش بینی های خود کار بسیار خوبی انجام می دهد.

طبق مشاهدات در استفاده از این ۲ طبقه‌بند، به طور کلی در اندازه لایه پنهان کمتر از ۶۴یا۳۲ در mlp، الگوریتم svm نتایج بسیار بهتر و قابل توجهی دارد(svm تقریبا با هر مقداری از هایپرپارامتر بهتر از mlp با پیچیدگی کم عمل میکند). اما با بالا بردن پیچیدگی و تعداد نورون‌های mlp و در اندازه لایه پنهان برابر ۶۴یا۳۲، کارایی این ۲ الگوریتم تقریبا نزدیک به هم بوده و برابر میباشد و این بدان معناست که hyperplane جداکننده تقریبا مشابهی پیدا میکنند.

از دیگر نتایج بدست آمده این است که در اجراهای مختلف svm بهینه‌ترین پارامترها از کرنل خطی و یا چندجمله‌ای با درجه پایین(و گاما پایین برای جلوگیری از overfitشدن) استفاده میکردند که دلیل این امر میتواند تعداد بالای نمونه‌های داده نسبت به ویژگی باشد.

در مرحله بعد مدل‌ها را بر روی کل دیتاست اعمال میکنیم و نتایج را مشاهده میکنیم:

	precision	recall	f1-score	support
1	0.66	0.55	0.60	15074
2	0.75	0.84	0.79	88956
3	0.77	0.65	0.71	52331
accuracy			0.75	156361
macro avg	0.73	0.68	0.70	156361
weighted avg	0.75	0.75	0.75	156361

Figure 3 گزارش طبقه بندی کلی mlp

	precision	recall	f1-score	support
1	0.70	0.47	0.56	15074
2	0.74	0.87	0.80	88956
3	0.79	0.62	0.70	52331
accuracy			0.75	156361
macro avg	0.74	0.66	0.69	156361
weighted avg	0.75	0.75	0.74	156361

Figure 4 گزارش طبقه بندی کلی svm

همانطور که انتظار میرفت نتایج برای کل داده نیز تکرار شدند و مقادیر بدست آمده مقادیر مناسبی برای مدل ما هستند.

مرحله‌ی سوم: یک مدل تحلیل‌پذیر

علت عملکرد خوب ورشهای مبتنی‌بر یادگیری عمیق ایجاد ویژگیهای پیچیده در لایه‌های پنهان شبکه است. یکی از مشکلات روشهای مبتنی‌بر شبکه‌های عصبی عدم تحلیل‌پذیری عملکرد آنها خصوصا در لایه‌ی پنهان میباشد. این به این معنی است که پیاده‌سازی آنها در محیط‌های مخاطره آمیز مانند مساله‌ی پیش روی ما ممکن نیست، زیرا به خاطر این عدم تحلیل‌پذیری امکان بررسی عملکرد مدل در تمام حالات وجود ندارد. به علاوه، این عدم تحلیل پذیری امکان بررسی دقیق اهمیت و تاثیر ویژگیها در حل مساله را نیز از ما خواهد گرفت. یکی از جایگزین‌های روش‌های مبتنی‌بر شبکه‌های عصبی استفاده از روش‌های مهندسی ویژگی است. مهندسی ویژگی شامل طراحی ویژگیهای مراتب بالاتر ۱۱و غیر خطی از ویژگیهای خام و سپس انتخاب آنها بر اساس عملکرد میباشد. به عنوان مثال، یکی از روشهای معمول مهندسی ویژگی استفاده از توابع اسکالر با فرم بسته (مانند توابع سینوسی، توابع نمایی، و توابع چندجلمهای) برای انتقال ویژگیها و بررسی عملکرد آنها روی داده‌های ارزیابی است. در این بخش، باید با استفاده از ابزار AutoFeatویژگیهای مناسب را برای مسالهی پیش رو بیابیم. این روش به طور اتوماتیک یک مجموعه ویژگی مناسب مهندسی کرده و به ما میدهد. مسائل از این دست که در آنها هدف طراحی اتوماتیک بخشی از پایپ لاین یادگیری ماشین است را AutoMLنامیده‌اند.

PCA به هیچ وجه یک روش انتخاب داده نیست بلکه transform داده ها از یک سیستم مختصات به سیستم دیگر است. یک اشتباه رایج که افراد مرتکب می شوند این است که PCA را برای متغیرهای غیر پیوسته به کار می برند. در حالی که از نظر فنی امکان استفاده از PCA بر روی متغیرهای گسسته یا متغیرهای طبقه‌بندی یا برچسب دار و حتی one-hot کدگذاری شده وجود ندارد. به بیان ساده، اگر متغیرها در یک صفحه مختصات قرار ندارند، PCA را برای آنها اعمال نمیکنیم. تنها راهی که PCA یک روش معتبر برای انتخاب ویژگی است این است که مهم ترین متغیرها آنهایی باشند که بیشترین واریانس را در آنها داشته باشند. که این امر معمولا اتفاق نمی‌افتد.

همانند PCA در LDA نیز ما یک بازنشانی از داده داریم و انتخاب ویژگی انجام نمیدهیم. ایده پشت LDA ساده است. از نظر ریاضی، ما باید یک فضای ویژگی جدید برای نمایش داده ها پیدا کنیم تا بتوانیم تفکیک کلاس ها را به حداکثر برسانیم.بدیهی است که اولین گام یافتن راهی برای اندازه گیری این ظرفیت جداسازی هر کاندیدای فضای ویژگی جدید است. فاصله بین میانگین‌های پیش‌بینی‌شده هر کلاس می‌تواند یکی از معیارها باشد، اما تنها این فاصله معیار خیلی خوبی نخواهد بود زیرا پراکندگی داده‌ها را در نظر نمی‌گیرد. در سال 1988، آماردانی به نام رونالد فیشر راه حل زیر را پیشنهاد کرد: حداکثر کردن تابعی که تفاوت بین میانگین ها را نشان می دهد، که با اندازه گیری تغییرپذیری درون کلاس نرمال می شود. پیشنهاد فیشر اساساً به حداکثر رساندن فاصله بین میانگین هر طبقه و به حداقل رساندن گسترش در خود کلاس است. بنابراین، ما به دو معیار می رسیم: درون طبقاتی و بین طبقاتی. با این حال، این فرمول تنها در صورتی امکان پذیر است که فرض کنیم مجموعه داده دارای توزیع نرمال است. این فرض ممکن است ارور به همراه داشته باشد زیرا اگر توزیع داده های به طور قابل توجهی غیر گاوسی باشد، LDA ممکن است خیلی خوب عمل نکند.

توضیحات بالا دلایلی بر عدم استفاده از pca و lda در این مساله بود. با توجه به این نتایج به کمک کتابخانه autofeat به انتخاب ویژگی از داده خام پرداخته و نتایج را بررسی میکنیم.

پس از اجرای الگوریتم،ویژگی‌های زیر باقی میمانند:

```
new_X.columns
```

```
Index(['position_t', 'position_s', 'position_o', 'roof_type_q', 'roof_type_x',
       'count_families', 'geo_level_1_id', 'area_percentage',
       'has_secondary_use', 'height_percentage', 'foundation_type_r',
       'foundation_type_i', 'foundation_type_w', 'other_floor_type_q',
       'other_floor_type_j', 'other_floor_type_s', 'ground_floor_type_f',
       'count_floors_pre_eq', 'ground_floor_type_m', 'ground_floor_type_v',
       'plan_configuration_a', 'plan_configuration_u', 'plan_configuration_c',
       'plan_configuration_q', 'has_secondary_use_other',
       'legal_ownership_status_w', 'has_secondary_use_rental',
       'land_surface_condition_n', 'legal_ownership_status_a',
       'land_surface_condition_o', 'has_superstructure_other',
       'legal_ownership_status_r', 'has_superstructure_timber',
       'has_superstructure_bamboo', 'has_superstructure_adobe_mud',
       'has_superstructure_stone_flag', 'has_superstructure_rc_engineered',
       'has_superstructure_mud_mortar_brick',
       'has_superstructure_mud_mortar_stone',
       'has_superstructure_rc_non_engineered',
       'has_superstructure_cement_mortar_brick',
       'has_superstructure_cement_mortar_stone'],
      dtype='object')
```

```
[28] new_X.shape
```

```
(260601, 42)
```

```
X.columns
```

```
Index(['geo_level_1_id', 'geo_level_2_id', 'geo_level_3_id',
       'count_floors_pre_eq', 'age', 'area_percentage', 'height_percentage',
       'has_superstructure_adobe_mud', 'has_superstructure_mud_mortar_stone',
       'has_superstructure_stone_flag',
       'has_superstructure_cement_mortar_stone',
       'has_superstructure_mud_mortar_brick',
       'has_superstructure_cement_mortar_brick', 'has_superstructure_timber',
       'has_superstructure_bamboo', 'has_superstructure_rc_non_engineered',
       'has_superstructure_rc_engineered', 'has_superstructure_other',
       'count_families', 'has_secondary_use', 'has_secondary_use_agriculture',
       'has_secondary_use_hotel', 'has_secondary_use_rental',
       'has_secondary_use_institution', 'has_secondary_use_school',
       'has_secondary_use_industry', 'has_secondary_use_health_post',
       'has_secondary_use_gov_office', 'has_secondary_use_use_police',
       'has_secondary_use_other', 'land_surface_condition_n',
       'land_surface_condition_o', 'land_surface_condition_t',
       'foundation_type_h', 'foundation_type_i', 'foundation_type_r',
       'foundation_type_u', 'foundation_type_w', 'roof_type_n', 'roof_type_q',
       'roof_type_x', 'ground_floor_type_f', 'ground_floor_type_m',
       'ground_floor_type_v', 'ground_floor_type_x', 'ground_floor_type_z',
       'other_floor_type_j', 'other_floor_type_q', 'other_floor_type_s',
       'other_floor_type_x', 'position_j', 'position_o', 'position_s',
       'position_t', 'plan_configuration_a', 'plan_configuration_c',
       'plan_configuration_d', 'plan_configuration_f', 'plan_configuration_m',
       'plan_configuration_n', 'plan_configuration_o', 'plan_configuration_q',
       'plan_configuration_s', 'plan_configuration_u',
       'legal_ownership_status_a', 'legal_ownership_status_r',
       'legal_ownership_status_v', 'legal_ownership_status_w'],
      dtype='object')
```

```
[29] X.shape
```

```
(260601, 68)
```

در حالیکه ویژگی‌های اولیه شامل مقادیر زیر هستند:

در یک بررسی اولیه میتوان مشاهده کرد که داده‌هایی که مشکل داشتند(در قسمت اول ذکر شد) حذف شده اند. برای مثال داده‌هایی که بازه بزرگتری نسبت به داده‌های دیگر داشتند و باعث ایجاد مشکل برای مدلسازی میشدند حذف شده‌اند مانند داده‌های geo2و geo3 .

همچنین با بررسی بیشتر میتوان حدس زد که داده‌های با همبستگی کم با پاسخ نیز از مجموعه حذف شده اند.

با این اوصاف پس از اجرای الگوریتم‌های logistic regression و svm به کمک این ویژگیها تنها مقدار اندکی بهبود در نتایج مدلسازی مشاهده میشود که نتایج مطلوبی نیستند. یکی از علل این امر میتواند به این دلیل باشد که در مراحل انتخاب ویژگی داده‌ها نرمالسازی نشده اند (طبق تست‌های مختلف که در طی مراحل انجام پروژه برای حل این مساله انجام گرفت ، نرمالسازی سهم زیاد و قابل توجهی در بهبود دقت یادگیری در این مساله داشت)