به نام خدا

هوش مصنوعی **پروژه چهارم**

علی دارابی – شماره دانشجویی : ۸۱۰۱۰۰۲۶۴

عنوان پروژه : Machine Learning

بررسی مجموعه داده

سوال ۱ :

در jupyter notebook انجام شده است.

سوال ۲ :

در jupyter notebook انجام شده است.

سوال ۳ :

نمودار correlation ویژگی ها، در jupyter notebook آورده شده است، باتوجه به این نمودار ، سه ویژگی زیر بیشترین مقدار وابستگی را به ستون هدف دارند :

MntCoffee, Income, MntMeatProducts

سوال ۴ :

در jupyter notebook انجام شده است.

سوال ۵ :

در jupyter notebook انجام شده است.

سوال ۶ :

برای اینکه دقت مدل های ما افزایش پیدا کند، داده های **Outlier** را حذف کردیم، همچنین برای اینکه بتوانیم نمودار correlation ویژگی هارا رسم کنیم، داده های خالی را با **میانگین** آن ستون پر کردیم. همچنین ویژگی های غیر عددی را به عدد تبدیل کردیم.

پیش پردازش مجموعه داده

سوال ۷ :

روش اول : پرکردن مقادیر خالی با یک عدد ثابت : **مزایا** : ساده و سریع است. **معایب** : اگر مقدار ثابت انتخابی ، تفاوت زیادی با مقادیر اصلی که حذف شده اند داشته باشد، میتواند باعث اختلال در فرایند مدلسازی شود.

روش دوم : پرکردن مقادیر خالی با میانگین یا میانه آن ستون : **مزایا** : چون از میانگین یا میانه آن ستون استفاده میکنیم، میتواند نمایشگر خوبی باشد و در فرایند مدل سازی، مشکلی برای ما ایجاد نکند. **معایب** : اگر داده های outlier داشته باشیم میتواند روی میانگین ما تاثیر بگذارد و میانگین دیگر نمایشگر خوبی برای داده های حذف شده نباشد.

روش سوم : پرکردن مقادیر خالی باتوجه به مقدار ردیف قبلی. **مزایا** : سریع و ساده است و داریم از خود دیتاست استفاده میکنیم و دیتا جدیدی اضافه نکردیم، درنتیجه مشکلی از نظر مدلسازی برنمیخوریم. **معایب** : اگر چندین خانه پشت سر هم خالی باشند و ما از یک مقدار برای پرکردن همه آنها استفاده کنیم، به مشکل میخوریم.

روش چهارم : استفاده از linear Interpolation. **مزایا** : نمایشگر خوبی برای داده ی حذف شده است زیرا که باتوجه به داده های قبلی و بعدی ساخته شده است. **معایب** : اگر دیتاست ما sort نشده باشد و از این روش استفاده کنیم، به مشکل برمیخوریم.

در نتیجه، از روش های گفته شده، استفاده از **میانگین** از باقی روش ها منطقی تر بنظر می آید که در کد هم از همین روش استفاده شده است.

سوال ۸ :

ویژگی های زیر بیشترین تعداد داده گمشده را داشتند :

Income = 223

MntCoffee = 205

MntGoldProds = 13

NumWebVisitsMonth = 200

سوال ۹ :

Normalizing ، تمامی داده ها را در بازه ۰ و ۱ می آورد و فرمول آن بدین شکل است:

X_norm = (X – X_min) / (X_max – X_min)

Standardizing تمامی داده ها طوری rescale میکند تا میانگین 0 و واریانس 1 داشته باشند و فرمول آن بدین شکل است:

X_Standard = (X – X_mean) / (X_std)

علت استفاده ما از این روش ها :

بسیاری از الگوریتم های یادگیری ماشین، وقتی داده های ما همگی در یک scale باشند، سریع تر کار میکنند و سریع تر همگرا میشوند و استاندارد سازی باعث میشود که ویژگی هایی که بازه های بزرگتری دارند، غالب نباشند.

استفاده از استاندارد سازی باعث میشود که تمام ویژگی ها حول صفر مرکزیت داده باشند و از اینکه ویژگی هایی که مقدار داده های آنها بزرگتر است غالب شوند، جلوگیری میکند.

نرمالایز کردن باعث میشود تاثیر داده های outlier ها که میتوانند توزیع مارا برهم بریزند، کمتر شود و تاثیر آنهارا کاهش میدهد.

در این پروژه هم ما نیاز به نرمالایز کردن داریم که در کد انجام شده است.

سوال ۱۰ :

برای چنین ویژگی هایی که عددی نیستند، ما باید در ابتدا آنها را با یک پیش پردازش مناسب، تبدیل به عدد کنیم. برای مثال در پروژه ما هم ویژگی های Education و Marital_Status ، بصورت string هستند و باید آنها را به صورت عددی دربیاریم. راه هایی که میتوان این پیش پردازش را انجام داد :

استفاده از روش باینری : برای هر string یک ویژگی جدید اضافه میکنیم و برای هر داده، اگر آن ویژگی را داشت ۱ و اگر نداشت ۰ میگذاریم.

استفاده از روش عددگذاری و کد کردن : در این روش برای هر string یک عدد در نظر میگیریم و در واقع کدینگ انجام میدهیم، برای مثال اگر ۳ نوع string داشته باشیم، میتوانیم آنها را به ۱ و ۲ و ۳ کد کنیم.

برای داده های دسته هایی که بصورت object و یا string هستند برای آنهایی که معتبر هستند، باید این پیش پردازش را انجام دهیم. و داده های غیر معتبر را میتوان حذف کرد.

در این پروژه هم ما برای ویژگی هایی که عددی نبودند، از روش کدینگ استفاده کردیم.

سوال ۱۱ :

بله، در صورتی که ستون یک ویژگی با ستون هدف وابستگی بسیار پایینی داشته باشد، بدین معناست که این ویژگی برای ساخت مدل کمکی به ما نمیکند و در واقع به ستون هدف نامرتبط است. در نتیجه میتوان آنها را به کلی حذف کرد.

سوال ۱۲ :

نسبت این تقسیم در اصل بستگی به تعداد داده ها در دیتاست ما دارد، برای مثال در دیتاست های نسبتا بزرگ مانند همین پروژه، نسبت (train)۸۰ به (test)۲۰ یا ۷۰ به ۳۰ مناسب است. در دیتاست های کوچکتر میتوان از نسبت های ۶۰ به ۴۰ هم استفاده کرد.

برای ساخت این دسته ها، روش های زیر را میتوان بکار برد :

داده های تست را از اول برداریم و بقیه را به عنوان داده train در نظر بگیریم.

داده های train را از اول برداریم و بقیه را به عنوان داده test در نظر بگیریم. برای مثال اگر داده های ما بر اساس زمان باشند این روش بهتر است.

بصورت رندوم و یا shuffle داده هارا انتخاب کنیم.

سوال ۱۳ :

از دسته ی validation ، برای اعتبارسنجی و ارزیابی هایپر پارامتر های مدل به کار میرود و در واقع در هر مرحله میتوان با داده های این دسته بررسی کرد که واقعا مدل ما بهبود یافته است یا نه. چون داده های این دسته متفاوت از داده های دسته ی تست هستند در واقع میتوانند به مدل ما کمک کنند تا در هر iteration عملکرد خود را بررسی کند تا بتواند در هرمرحله پیشرفت کند. برای مثال، میتوان داده ها را به نسبت ۷۰ به ۱۵ به ۱۵ برای train و test و validation تقسیم بندی کنیم.

سوال ۱۴ :

K-fold cross validation برای ارزیابی مدل هایی استفاده میشود که روی دیتاست کوچکی مدلسازی شده اند. بدین صورت کار میکند که در ابتدا دیتاست را به k تکه برابر تقسیم میکند و یک تکه را برای validation و $k-1$ تکه را برای train برمیدارد و این مرحله را k بار انجام میدهد، و در هر تکرار یک تکه دیگر از k تکه را برای validation ، انتخاب میکند. در هر تکرار مدل را بویسیله ی داده های validation ارزیابی میکند و در نهایت میانگین این k تکرار را به عنوان ارزیابی نهایی از عملکرد مدل در نظر میگیرد.

فاز اول : Linear Regression

سوال ۱۵ :

هدف ما در اینجا پیدا کردن معادله خطی است که بین داده های ستون هدف و ستون ویژگی دلخواه بهترین فیت است. برای اینکار باید مقدار RSS را کمینه کنیم، برای کمینه کردن آن ، از آن نسبت به آلفا و بتا مشتق میگیریم و برابر با صفر میگذاریم و با حل کردن معادله های بدست آمده ، میتوانیم آلفا و بتا را بدست بیاوریم.

سوال ۱۶ :

بهترین ویژگی برای انتخاب MntCoffee است.

علت انتخاب این آن است که MntCoffee بیشترین مقدار وابستگی (correlation) را با ستون هدف دارد.

سوال ۱۷ :

متد RSS :

RSS برابر است با سیگمای actual y value ها منهای predicted y ها به توان ۲. با مینیمایز کردن مقدار RSS ما میتوانیم بهترین معادله خط قابل فیت کردن را پیدا کنیم. زیرا که با منیمایز کردن RSS اختلاف مقادیر واقعی با مقادیر پیشبینی شده به حداقل میرسد. و نوعی معیار برای ارزیابی خط فیت شده است.

متد MSE :

MSE یا Mean Squared Error برابر با RSS تقسیم بر n (تعداد داده ها) است. و در واقع میتوان گفت MSE میانگین خطاهای مقادیر واقعی و مقدار پیشبینی شده است. و همانند RSS با مینیمایز کردن این مقدار میتوان به بهترین معادله خط قابل فیت کردن رسید. و نوعی معیار برای ارزیابی خط فیت شده است.

متد RMSE :

RMSE یا Root Mean Square Error برابر با ریشه ی MSE است. RMSE هم همانند MSE نوعی معیار برای ارزیابی مدل ما است و هرچه مقدار آن کوچکتر باشد نشان دهنده این است که پیشبینی های ما به مقادیر واقعی نزدیک تر هستند.

متد R2 Score :

RMSE مرزی ندارد، بنابراین تعیین اینکه آیا یک مقدار RMSE واقعا مقدار خوبی است یا نه، دشوار است. به همین دلیل از R2 Score استفاده میکنیم. R2 Score ، با مقایسه ی مقادیر سیگمای مربع مقدار واقعی منهای مقدار پیشبینی شده با مقدار سیگمای مربع مقدار واقعی منهای مقدار میانگین، کارایی معادله خط فیت شده را بررسی میکند و هر چه که مقدار آن به ۱ نزدیک تر باشد نشان دهنده ی این است که خط فیت شده ی ما تعداد داده بیشتری را به درستی پیشبینی کرده است.

سوال ۱۸ :

این متد هارا روی ۴ ویژگی که بیشترین وابستگی را با ستون هدف داشتند انجام دادیم. همانطور که در قسمت قبل هم گفته شد، هرچه که مقدار RMSE به صفر نزدیک تر باشد، بیانگر این است که آن خط، فیت بهتری است. در مقادیر بدست آمده در **jupyter notebook** هم این موضوع قابل مشاهده است. R2 Score برای یک ویژگی هرچقدر به ۱ نزدیک تر باشد بیانگر این است که آن خط فیت بهتری است. در **jupyter notebook** هم این موضوع قابل مشاهده است. و همانطور که انتظار داشتیم ویژگی که وابستگی بیشتری با ستون هدف داشت، RMSE کمتر و R2 Score بالاتری داشت.

فاز سوم : طبقه بندی

سوال ۱۹ :

Confusion Matrix هارا در jupyter notebook رسم شده اند.

مقایسه : طبق نتایج بدست آمده از روی confusion matrix های هر مدل، میتوان گفت که مدل ساخته شده Decision Tree بالاترین دقت را بین سه مدل ساخته شده دارد. در رتبه دوم، Logistic Regression قرار دارد و مقدار دقت آن مقداری کمتری از Decision Tree است. در رتبه ی آخر K-Neighbors قرار دارد و میزان دقت آن اختلاف زیادی با دو مدل قبلی دارد.

سوال ۲۰ :

در jupyter notebook انجام شده است.

سوال ۲۱ :

Overfitting: بدین معناست که مدل ساخته ی شده ما بیش از حد به داده‌های train وابسته شده و نمی‌تواند روی داده‌های جدید عملکرد خوبی داشته باشد. این پدیده معمولا زمانی رخ می‌دهد که مدل خیلی پیچیده باشد و سعی کند نویزهای داده‌های train را یاد بگیرد.

Underfitting: یعنی مدل نتوانسته روابط و الگوهای مهم در داده‌ها را به درستی یاد بگیرد. این پدیده معمولا زمانی اتفاق می‌افتد که مدل خیلی ساده باشد و انعطاف‌پذیری کافی برای مدل‌سازی داده‌ها را نداشته باشد.

این پدیده ها در کد بررسی شده.

سوال ۲۲ :

تاثیر حذف کردن یا نکردن داده های outlier :

وقتی داده های outlier را دوباره به مجموعه دیتاست اضافه کردیم، دقت مدل های ساخته شده کمی کاهش پیدا کرد.

تاثیر نرمالایز کردن یا نکردن :

وقتی بدون نرمالایز کردن دیتاست مدل هارا train کردیم، در نهایت دقت مدل ها کمی کاهش پیدا کرد.

تاثیر حذف کردن یا نکردن داده های بی معنی :

وقتی داده های بی معنی را حذف نکنیم، تاثیر بسیار کمی در دقت نهایی مدل های ما میگذارد.

سوال ۲۳ :

در jupyter notebook انجام شده است.

سوال ۲۴ :

n_estimators: بیانگر تعداد درختان است. درختان بیشتر واریانس را کاهش می دهد اما باعث افزایش زمان train میشود. مقدار معقول برای این پارامتر بین ۱۰۰ تا ۵۰۰ است.

max_depth: بیانگر حداکثر عمق هر درخت است. overfitting را کنترل می کند. مقادیر معقول آن بین ۵ تا ۱۰ متغیر است و درختان عمیق تر در مقابل overfitting مقاوم تر هستند.

min_samples_split: بیانگر حداقل sample های مورد نیاز برای تقسیم یک گره است. مقادیر معقول آن بین ۲ تا ۱۰ است. و مقادیر بالاتر از overfitting جلوگیری میکند.

بررسی تاثیر این پارامتر ها در در jupyter notebook انجام شده است.

سوال ۲۵ :

Bias: بیانگر این است که پیش‌بینی‌های مدل چقدر از مقدار واقعی دور یا غلط است. و bias زیاد به این معنی است که پیش‌بینی‌های مدل با مقدار صحیح فاصله دارند.

Variance: بیانگر این است که چقدر پیش بینی های مدل بین مجموعه های train مختلف متفاوت است. واریانس بالا به این معنی است که پیش بینی ها بر اساس تغییرات کوچک در داده ها به طور قابل توجهی تغییر می کنند.

یک trade off بین bias و واریانس وجود دارد، یعنی معمولا با کاهش یکی از آنها سبب افزایش دیگری میشود.

با مقایسه ی نتایج حاصل از Decision Tree و Random Forest، مشاهده میشود که مدل RF، Bias و Variance کمتری نسبت به DT دارد. بنظر میرسد که RF با میانگین گیری از تعداد زیادی درخت تصمیم، Bias را کاهش داده است. همچنین بنظر میرسد که میانگین گیری از درخت های تصمیم متعدد، مقدار واریانس هم کاهش داده است. به طور کلی، میتوان گفت که مدل RF تعادل بهتری بین bias و واریانس در مقایسه با یک درخت تصمیم به دست می‌آورد.

نتایجی که ما در کد بدست آوردیم هم با نتایج تئوری مطابقت دارد، بدین معنا که بطور کلی RF ، bias و واریانس کمتری نسبت به یک درخت تصمیم داشت و دقت آن بیشتر بود.

سوال ۲۶ :

افزودن نویز به داده های موجود در ردیف های مختلف یک دیتاست، می تواند تاثیر مثبتی بر حفظ امنیت و حریم شخصی افراد داشته باشد. زیرا که اینکار باعث می شود که شناسایی افراد از روی داده ها دشوارتر شود. البته باید در نظر داشت که افزودن نویز به داده ها می تواند تاثیر منفی هم داشته باشد. به عنوان مثال، اگر نویز خیلی زیاد باشد، ممکن است باعث شود که داده ها غیرقابل استفاده شوند.

سوال ۲۷ :

چون که نویز لاپلاس با استفاده از توزیع لاپلاس نویز را میسازد، باعث میشود داده ها بیشتر تغییر کنند و مبحث حفظ امنیت و حریم خصوصی بیشتر رعایت میشود اما از طرفی باعث میشود که داده های ما به مشکل بخورند و احتمالا دقت ما کاهش یابد. نویز نمایی از توزیع نمایی برای ساخت نویز استفاده میکند و مبحث حریم خصوصی و حفظ امنیت کمتر رعایت میشود اما داده ها تقریبا سالم باقی میمانند یعنی دقت ما کاهش زیادی نمیابد.

سوال ۲۸ :

از نویز لاپلاس استفاده کردیم و همانطور که انتظار میرفت، دقت تمامی مدل های ما مقداری کاهش یافت.

نتایج در jupyter notebook موجود است.

سوال ۲۹ :

Gradient Boosting ، یک تکنیک یادگیری ماشین برای مسائل رگرسیون و طبقه بندی است. که بر اساس درخت های تصمیم گیری است. روش کار آن بدین صورت است در ابتدا یک مدل ساده مانند یک درخت تصمیم را روی دیتاست train میکند و سپس مقدار اختلاف یا ارور این مدل را محاسبه میکند. سپس یک مدل جدید (درخت تصمیم) برای پیداکردن ارور های مدل قبلی train میکند و این مدل تلاش میکند تا درخت قبلی را بهبود ببخشد. سپس این مدل به مجموعه مدل ها اضافه میشود و پیشبینی های آنها توسط میانگین وزن دار ترکیب میشود و تمرکز روی این است که loss function مینیمایز شود. این مراحل آنقدر تکرار میشوند تا به شرط خاتمه (مانند تعداد تکرار یا یک مقدار حدی برای کمینه خطا) برسیم.

سوال ۳۰ :

XGBoost یک gradient boosting library بهینه شده است که برای آموزش کارآمد و مقیاس پذیر مدل های یادگیری ماشین طراحی شده است. روش کار آن بدین صورت است که پیش‌بینی‌های چند مدل ضعیف را برای تولید یک پیش‌بینی قوی تر ترکیب می‌کند. و به دلیل توانایی آن در مدیریت مجموعه داده های بزرگ ، به یکی از محبوب ترین و پرکاربردترین الگوریتم های یادگیری ماشین تبدیل شده است. XGBoost به طور موثر مقادیر از دست رفته را کنترل می کند.

XGBoost از مجموعه ای از درختان تصمیم به عنوان مدل های پایه استفاده میکند. هر درخت بر روی ارور های درخت های قبلی آموزش داده میشود. بنابراین اولین درخت بر روی داده های train اصلی آموزش داده میشود. دومین درخت بر روی ارور های درخت اول train داده میشود و به همین ترتیب ادامه پیدا میکند.

هنگام آموزش یک درخت، XGBoost از الگوریتم greedy برای تقسیم داده ها در هر گره استفاده میکند و بهترین تقسیم که باعث بهبود حداکثری تابع هدف میشود را پیدا میکند. هر درخت از ارور های درخت های قبلی آموزش داده شده یاد میگیرد. و پیشبینی های تمام درخت ها از طریق میانگین گیری وزن دار با هم ترکیب میشوند تا پیشبینی نهایی انجام شود. بنابراین هر درخت متوالی سعی میکند درخت های قبلی را بهبود ببخشد. و هرچه تعداد درخت ها بیشتر باشد، مدل قویتر و دقیق تر میشود. XGBoost همچنین دارای پارامترهای منظم سازی است تا overfitt شدن را بویسیله ی افزایش تعداد درخت ها کنترل کند.

سوال ۳۱ :

در jupyter notebook انجام شده است.
