پروژه دوم درس یادگیری ماشین

Flight Delay

استاد: دکتر ساجدی

دانشجو: ارشیا یوسفی نژاد (610302085)

Dataset 1
Feature Engineering 1
Drop some columns 1
Add New Features 1
Splitting data to train and test (like time series data) 2
Splitting data: 2
Target feature: 2
Data Preprocessing 2
Models 3
MLP 3
Evaluate MLP 3
ELM (Extreme Machine Learning) 4
Resampling data 4
Evaluate ELM 4
Auto Encoder 5
Evaluate Auto Encoder 5

Dataset

دو دیتاست برای سال های 2019 و 2020 بودند برای آنکه تمایز بین زمان ها در نظر گرفته شده باشد ستونی تحت عنوان year را اضافه شده.

Feature Engineering

Drop some columns

با توجه به تحلیل هایی که روی دیتاست صورت گرفته (تحلیل هایی که در کد های خود سایت kaggle موجود بود) یکسری از ستون ها را حذف کردیم (ستونهای که بیشتر مربوط به ID و امثالهم میشدند که بار اطلاعاتی لازمه را نداشتند)

Add New Features

برای اینکه مدل عملکرد خوبی داشته باشد یکسری ویژگی اضافه شده
ویژگی DISTANCE_cat: ویژگی DISTANCE را به چهار قسمت تقسیم کردیم (همون چارک در بحث های آماری) و ویژگی بدست آمده حاوی چهار نوع دسته بند هست (چهار نوع categort)

ویژگی ARR_TIME_BLK: در این ویژگی زمان ها را دسته بندی کردیم. (به عنوان مثال ساعت 6:00 تا 6:59، ...، 23:00 تا 23:59)

ویژگی count_dep_time_blk: این ویژگی برابر تعداد تاخیر های حرکت کردن پرواز در هر بازه ی زمانی است. (به عنوان مثال در بازه ی زمانی 6:00 تا 6:59 به تعداد 5436)

ویژگی count_arr_time_blk: این ویژگی برابر تعداد تاخیر های رسیدن پرواز در هر بازه ی زمانی است. (به عنوان مثال در بازه ی زمانی 7:00 تا 7:59 به تعداد 40653)

Handling missing values

حذف داده هایی که شامل missing value بودن. پیش پردازشی روی این قسمت انجام نشده، زیرا دادگان به اندازه ی کافی زیاد بوده اند.

Data Preparation

Splitting data to train and test (like time series data)

این نوع دادگان شبیه به دادگان time series هستند، یعنی اینکه یکسری داده از قبل داریم و میخوایم پیش بینی کنیم با توجه به این داده ها آیا پرواز پیش رو تاخیر دارد یا نه.
برای این امر یک ستون با نام DEP_TIME_hours ایجاد کردم که این ستون داده هایی که در ستون هایی ARR_TIME و DEP_TIME هستند را بصورت ساعتی نشان میدهد (به عنوان مثال داده هایی که بصورت 0600 هستند را بصورت 06:00 نمایش میدهد)
و همچنین ستون DEPARTURE_DATETIME ، که این ستون زمان ها را در کنار هم قرار می گذارد (برای فرآیند sort-ای که میخواهیم انجام دهیم چون فرض بر این گرفته شده داده ها بصورت سری زمانی هستند)

Splitting data:

هشتاد درصد داده هایی ابتدایی به train تعلق میگیرد و بیست درصد انتهایی به test.

Target feature:

اگر پرواز دیر راه بیفتد یا دیر برسد یک درنظر گرفته میشود در غیر اینصورت صفر

Data Preprocessing

ویژگی های category و numeric را جدا کرده و روی ویژگی categorical عمل onehotencoding و روی ویژگی numerical عمل normalization صورت گرفته.

Models

MLP

معماری ای که برای Multi-layer perceptron انتخاب کردم دارای 5 hidden layer است. و به لایه ی ابتدایی ترم l1 regulariztion رو افزودم (تا کمی از overfit شدن مدل جلوگیری کرده باشم). همچنین Early stopping هم به این مدل اضافه شده تا هم از overfit شدن جلوگیری بشه و هم از طولانی شدن زمان اجرا.

Optimizer: ‘adam’ loss=’binary_crossentropy’ metrics=’accuracy’

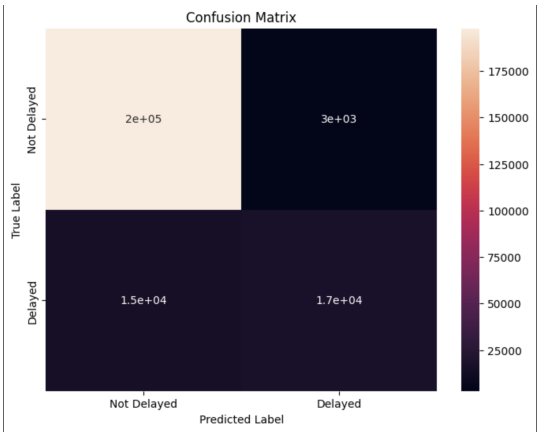
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 150)               13500

 dense_1 (Dense)             (None, 100)               15100

 dense_2 (Dense)             (None, 50)                5050

 dense_3 (Dense)             (None, 10)                510

 dense_4 (Dense)             (None, 1)                 11

=================================================================
Total params: 34171 (133.48 KB)
Trainable params: 34171 (133.48 KB)
```

معماری مدل:
این مدل را هم دیپ در نظر گرفتم. البته روی یک لایه هم امتحان شد ولی نتیجه ی مطلوبی نداشت!
برای پیاده سازی این معماری از Keras استفاده شده.

Evaluate MLP

Loss: 0.25365662574768066, Accuracy: 0.9233921766281128

	precision	recall	f1-score	support
0.0	0.93	0.99	0.96	200939
1.0	0.85	0.54	0.66	32092
accuracy			0.92	233031
macro avg	0.89	0.76	0.81	233031
weighted avg	0.92	0.92	0.92	233031



ELM (Extreme Machine Learning)

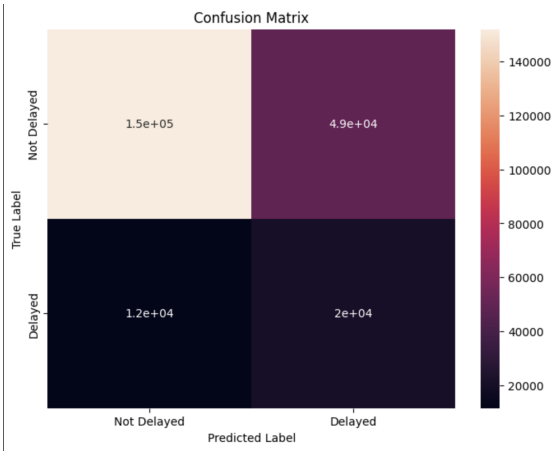
Resampling data

با توجه به آنالیز صورت گرفته روی داده ها؛ دادگان imbalace بوده اند، یا کمک پکیج imblearn داده ها را به حالت balace شده در می آوریم. از روش undersapeling استفاده شده، که برای بالانس شدن تعداد داده با برچسب صفر که تعدادشان بسیار بیشتر از دادگان با برچسب یک بوده را کاهش دادیم تا دیتاست به حال بالانس شده در آید.

معماری این مدل را با کمک کلاس ELM ساختیم. ورودی های کلاس شامل hidden_dim، input dim است.
توابعی که در این کلاس تعریف کردیم:
Apply_activation: این تابع باعث میشه که وزن ها غیر خطی شوند. و از دو activation استفاده شده در آن relu و sigmoid.
Train_network: این تابع محاسبات مربوط به الگوریتم ELM صورت میگیرد. یعنی وزنها مقدار دهی میشوند
Make_predictions : از این تابع هم برای پیش بینی و ارزیابی استفاده میشود. یعنی از وزنهای آپدیت شده استفاده میکنیم.

Evaluate ELM

	precision	recall	f1-score	support
0.0	0.93	0.76	0.83	200939
1.0	0.29	0.64	0.40	32092
accuracy			0.74	233031
macro avg	0.61	0.70	0.62	233031
weighted avg	0.84	0.74	0.77	233031



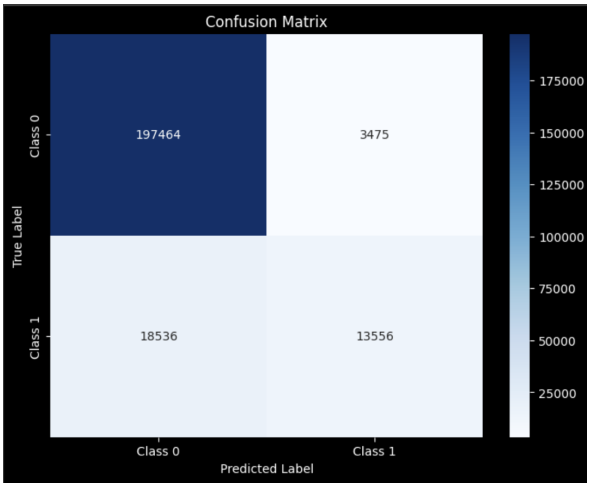
Auto Encoder

ابتدا دادگان X را به autoencoder داده و خروجی آنها که X_reconstruct شده است را به مدل MLP میدهیم. میدانیم که یکی از کاربرد های اصلی autoencoder ها حذف نویز است.
خروجی این مدل را به مدل MLP میدهیم. همانطور که ملاحظه می شود نتایج ضعیف تری نسبت به داده هایی اصلی دارد (زیرا در اینجا ما داده های reconstruct شده را به ورودی MLP دادیم)
همانطور هم که در مثال عکس در کلاس گفته شده بود، اگر تصویری به این مدل دهیم، این مدل کیفیت تصویر اصلی را نخواهد داشت (البته در برخی موارد ممکن است کیفیت را بالاتر ببرد مثل تصاویر قدیمی که کیفیتشان پایین است)

معماری این مدل:
در این مدل از پکیج keras استفده شده.

```
input_layer = Input(shape=(actual_dim,))

encoded = Dense(512, activation='relu')(input_layer)
encoded = Dense(encoding_dim, activation='relu')(encoded)

decoded = Dense(512, activation='relu')(encoded)
decoded = Dense(actual_dim, activation='sigmoid')(decoded)

autoencoder = Model(input_layer, decoded)
encoder = Model(input_layer, encoded)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Evaluate Auto Encoder




```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.91      0.98      0.95    200939
         1.0       0.80      0.42      0.55     32092

    accuracy                           0.91    233031
   macro avg       0.86      0.70      0.75    233031
weighted avg       0.90      0.91      0.89    233031
```