در ابتدا درباره دیتاست صحبت کنیم:

ID,
Delivery_person_ID,
Delivery_person_Age,
Delivery_person_Ratings,
Restaurant_latitude,
Restaurant_longitude,
Delivery_location_latitude,
Delivery_location_longitude,
Type_of_order,
Type_of_vehicle,
Time_taken(min)

در ابتدا که برای هر سطر یک آیدی اختصاص یافته داریم
سپس شخصی که غذارا تحویل می دهد یک آي دی دارد
سپس سن شخصی که غذارا تحویل می دهد یک آي دی دارد
سپس امتیازدهی به شخصی که غذارا تحویل می دهد یک آي دی دارد
عرض جغرافیایی یک رستواران
طول جغرافیایی یک رستوران
محل رسیدن عرض جغرافیایی
محل رسیدن طول جغرافیایی
به طور خلاصه طول و عرض جغرافیایی مبدا و مقصد را داریم
نوع سفارش
نوع وسیله
مقدار زمان سپری شده

B379,
BANGRES18DEL02,
34,
4.5,
12.913041,
77.683237,
13.043041,
77.813237,
Snack,
scooter,
33

این پروژه برای دو شرکت بزرگ غذا هندی ها است
خدمات تحویل غذا مانند Zomato و Swiggy باید زمان دقیق تحویل سفارش شما را نشان دهند تا شفافیت را با مشتریان خود حفظ کنند.
این شرکت‌ها از الگوریتم‌های یادگیری ماشینی برای پیش‌بینی زمان تحویل غذا بر اساس مدت زمانی که شرکت های قبلی برای همان مسافت در گذشته صرف کرده‌اند، استفاده می‌کنند.

برای پیش بینی زمان تحویل غذا در زمان واقعی، باید فاصله بین نقطه آماده سازی غذا و نقطه مصرف غذا را محاسبه کنیم. پس از یافتن فاصله بین رستوران و مکان‌های تحویل، باید رابطه‌ای بین مدت زمانی که قبلا رستوران ها یا شرکت ها برای تحویل غذا در گذشته برای همان فاصله صرف کرده‌اند، پیدا کنیم.

بنابراین، برای این کار، به مجموعه داده‌ای حاوی داده‌هایی که در بالا اشاره شد نیاز داریم.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

df = pd.read_csv("deliverytime.txt")
df.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Type_of_order	Type_of_vehicle	Time_taken(min)
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	75.912471	Snack	motorcycle	24
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	77.813237	Snack	scooter	33
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	77.688400	Drinks	motorcycle	26
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	77.026494	Buffet	motorcycle	21
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	80.289982	Snack	scooter	30

1. این پروژه استفاده از الگوریتم های یادگیری ماشین برای پیش بینی زمان تحویل غذا به منظور افزایش رضایت مشتری از خدمات تحویل غذا است.
کتابخانه های مورد نیاز را صدا میزنیم:
2. **بخش import**: این بخش کتابخانه های مختلف پایتون لازم برای پروژه مانند "matplotlib" ،"numpy" ،"pandas" و "seaborn" را وارد می کند. اینها کتابخانه هایی هستند که برای دستکاری، تحلیل و رسم داده ها استفاده می شوند.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           45593 non-null  object
 1   Delivery_person_ID           45593 non-null  object
 2   Delivery_person_Age          45593 non-null  int64
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object
 9   Type_of_vehicle              45593 non-null  object
 10  Time_taken(min)              45593 non-null  int64
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

```
[ ] df.isnull().sum()
```

```
ID                           0
Delivery_person_ID           0
Delivery_person_Age          0
Delivery_person_Ratings      0
Restaurant_latitude          0
Restaurant_longitude         0
Delivery_location_latitude   0
Delivery_location_longitude  0
Type_of_order                0
Type_of_vehicle              0
Time_taken(min)              0
dtype: int64
```

3. در این بخش همانطور که از خروجی نیز پیداست اطلاعاتی دررابطه با فایل txt بدست می آوریم.
با استفاده از df.info() متوجه میشویم که یازده ستون داریم (همان آی دی و طول و عرض جغرافیایی و...) و ۴۵۵۹۳ ورودی داریم که دیتاتایپ های آنها عبارت است از:
اعداد اعشاری(۵ ستون)
اعداد صحیح(۲ ستون)
object(چهار ستون) در واقع ترکیبی از حرف و عدد
همچنین اینفو تعداد سطرهای غیرتهی برای هر سطر را نشان می دهد مثلا عرض جغرافیایی همه سطرهارا داریم برای همین عدد ۴۵۵۹۳ را نمایش می دهد.

```
df.isnull().sum()
```

4. همانطور که مشخص است این خط کد هم تعداد سطرهایی که خالی اند را نمایش میدهد.

5. محاسبه فاصله بین دو طول و عرض جغرافیایی

مجموعه داده هیچ ویژگی ندارد که تفاوت بین رستوران و محل تحویل را نشان دهد. تنها چیزی که ما داریم، عرض و طول جغرافیایی رستوران و محل تحویل است. ما می توانیم از فرمول هارسین برای محاسبه فاصله بین دو مکان بر اساس طول و عرض جغرافیایی آنها استفاده کنیم.

در زیر نحوه یافتن فاصله بین رستوران و محل تحویل بر اساس طول و عرض جغرافیایی با استفاده از فرمول هارسین آمده است.

```
[ ]  # Set the earth's radius (in kilometers)
     R = 6371

     # Convert degrees to radians
     def deg_to_rad(degrees):
         return degrees * (np.pi/180)

     # Function to calculate the distance between two points using the haversine formula
     def distcalculate(lat1, lon1, lat2, lon2):
         d_lat = deg_to_rad(lat2-lat1)
         d_lon = deg_to_rad(lon2-lon1)
         a = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1)) * np.cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
         c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
         return R * c
```

```
     # Calculate the distance between each pair of points
     df['distance'] = np.nan

     for i in range(len(df)):
         df.loc[i,'distance'] = distcalculate(df.loc[i, 'Restaurant_latitude'],
                                              df.loc[i, 'Restaurant_longitude'],
                                              df.loc[i, 'Delivery_location_latitude'],
                                              df.loc[i, 'Delivery_location_longitude'])
```

فاصله بین دو نقطه از سطح زمین را با استفاده از فرمول هارسین محاسبه می کند.
شعاع زمین 6371 کیلومتر تعیین شده است.
تابع deg_to_rad برای تبدیل درجه به رادیان با ضرب درجه ورودی در π/180 تعریف شده است.
تابع distcalculate چهار پارامتر دارد: طول و عرض جغرافیایی دو نقطه. سپس با استفاده از فرمول هاورسین که بر اساس هندسه کروی است، فاصله بین این دو نقطه را محاسبه می کند.
در داخل تابع distcalculate، تفاوت های طول و عرض جغرافیایی بین دو نقطه با استفاده از تابع deg_to_rad از درجه به رادیان تبدیل می شود.
سپس فرمول هاورسین برای محاسبه فاصله بین دو نقطه با استفاده از فرمول اعمال می شود:

$$a = \sin^2(\Delta lat/2) + \cos(lat1) * \cos(lat2) * \sin^2(\Delta lon/2$$

$$c = 2 * \arctan2(\sqrt{a}، \sqrt{1-a}$$

فاصله = R * c، که در آن R شعاع زمین است.

کلا اینکه هاورسین چطور کار میکنه:

فرمول هاورسین یک فرمول ریاضی است که برای محاسبه فاصله بین دو نقطه از سطح یک کره مانند زمین استفاده می شود.
معمولاً در برنامه های ناوبری و مکان یابی جغرافیایی برای تعیین فاصله بین دو مجموعه مختصات جغرافیایی (طول و عرض جغرافیایی) استفاده می شود.

فرمول به شرح زیر است:

$$a = \sin^2(\Delta lat/2) + \cos(lat1) * \cos(lat2) * \sin^2(\Delta lon/2)$$
$$c = 2 * \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R * c$$

جایی که:
- lat1 و lon1 طول و عرض جغرافیایی نقطه اول هستند
- lat2 و lon2 طول و عرض جغرافیایی نقطه دوم هستند
- Δlat و Δlon اختلاف عرض و طول جغرافیایی بین دو نقطه هستند
- R شعاع زمین است (میانگین شعاع = 6371 کیلومتر)
- d فاصله بین دو نقطه در واحدهای مشابه با شعاع زمین است

حاصل فرمول هاورسین فاصله دایره بزرگ بین دو نقطه است که کمترین فاصله بین آنها در امتداد سطح کره است.

حالا که می دونیم هاورسین چیکار میکنه بهتره یه نگاهی به کد بندازیم
در ابتدا شعاع کره زمین رو تعریف کردیم
سپس اومدیم تابعی برای تبدیل درجه به رادیان تعریف کردیم
که درجه رو در پی/۱۸۰ ضرب می کنه
سپس تابع distcalculate رو تعریف کردیم که دقیقا داره مثل عملی که پله به پله تو هاورسین گفتیم رو پیاده سازی می کنه.
حالا که درجه رو تبدیل به رادیان کردیم تا بتونیم با استفاده از هاورسین فاصله بین رستوران تا محل تحویل رو به دست بیاریم یک ستون با عنوان distance به دیتاست اضافه می کنیم که فاصله برای محل تحویل تا مبدا را برای هر سطر محاسبه کرده(با استفاده از حلقه for در آن)

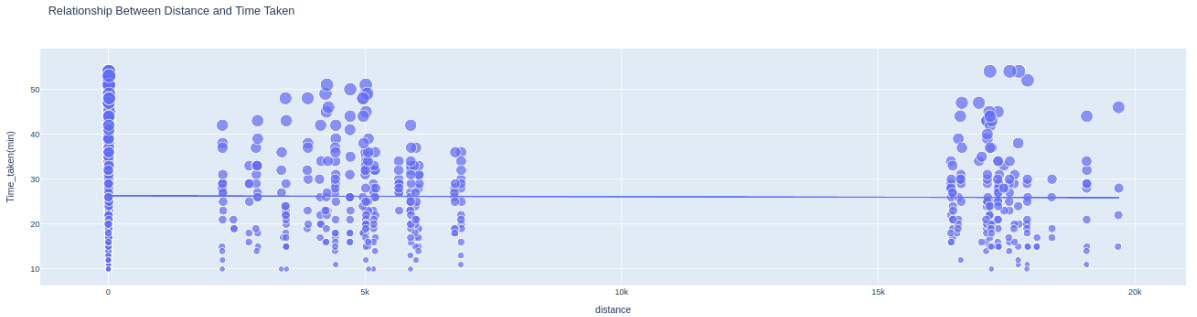
کاوش داده ها

حالا بیایید داده ها را بررسی کنیم تا روابط بین ویژگی ها را پیدا کنیم. با بررسی رابطه بین فاصله و زمان صرف شده برای تحویل غذا شروع می شود.

1.

```
figure = px.scatter(data_frame = df,
                    x = 'distance',
                    y = 'Time_taken(min)',
                    size = 'Time_taken(min)',
                    trendline = 'ols',
                    title = "Relationship Between Distance and Time Taken")

figure.show()
```

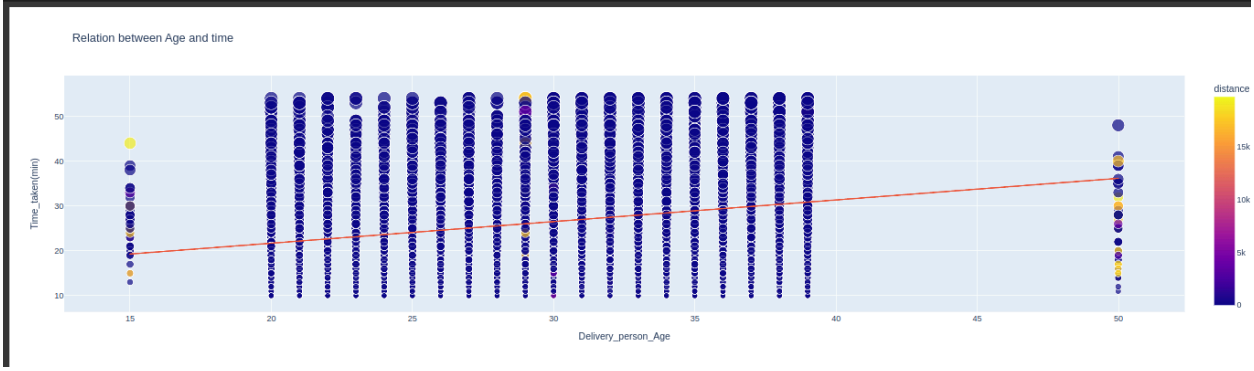


یک نمودار پراکندگی با عنوان "رابطه بین فاصله و زمان گرفته شده" را نشان می دهد. محور افقی (محور x) دارای برچسب "فاصله" است، در حالی که محور عمودی (محور y) برچسب "زمان گرفته شده" است. نمودار مجموعه ای از نقاط پراکنده در نمودار را نشان می دهد که مجموعه داده مقادیر را برای مسافت و زمان صرف شده نشان می دهد. این نمودار برای تجزیه و تحلیل بصری رابطه یا همبستگی بین دو متغیر عددی استفاده می شود: فاصله و زمان صرف شده برای آن فاصله.
یک رابطه ثابت بین زمان صرف شده و مسافت طی شده برای تحویل غذا وجود دارد. این بدان معناست که اکثر شرکتای تحویل غذا بدون توجه به مسافت، ظرف 25 تا 30 دقیقه غذا را تحویل می دهند.

2.

```
figure = px.scatter(data_frame = df,
                    x = 'Delivery_person_Age',
                    y = 'Time_taken(min)',
                    size = 'Time_taken(min)',
                    color = 'distance',
                    trendline = 'ols',
                    title = 'Relation between Age and time')

figure.show()
```



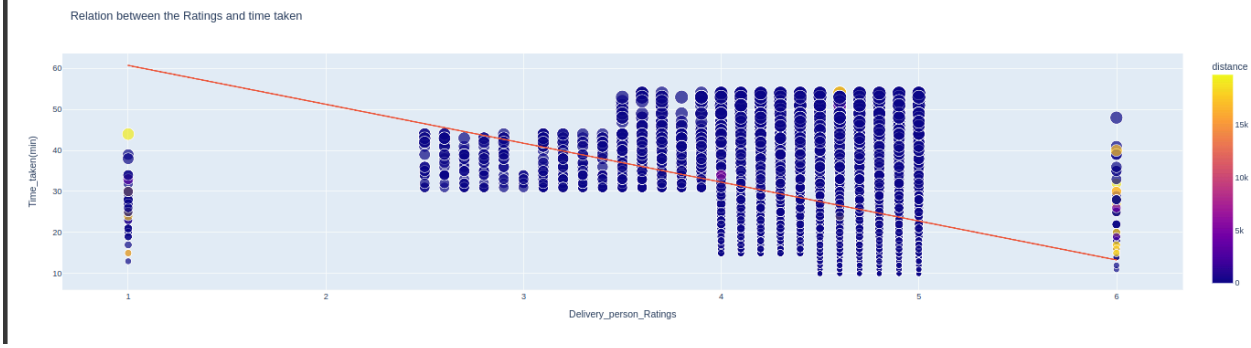
- محور x دارای برچسب "سن_تحویل دهنده" است که از حدود 27 تا 69 متغیر است.
- نقاط روی نمودار پراکندگی بر اساس مقدار "فاصله" آنها رنگ می شوند که با نوار رنگ در سمت راست نشان داده شده است. گرادیان رنگ از بنفش (مقادیر پایین تر) به زرد (مقادیر بالاتر) تغییر می کند.
- خط قرمز نشان دهنده یک خط رگرسیون خطی است که رابطه کلی بین سن و تاخیر را نشان می دهد.

روند کلی پیشنهاد شده توسط خط قرمز نشان می دهد که با افزایش "سن_تحویل"، "تأخیر_تحویل" نیز اندکی افزایش می یابد، به این معنی که سن بالاتر ممکن است با تاخیر بیشتر در این مجموعه داده همراه باشد. کدگذاری رنگ بر اساس فاصله می تواند نشان دهنده یک بعد اضافی مانند فاصله جغرافیایی یا یک متریک مرتبط باشد که با متغیرهای سن و تاخیر تغییر می کند.

3.

```
figure = px.scatter(data_frame = df,
                    x='Delivery_person_Ratings',
                    y = 'Time_taken(min)',
                    size = 'Time_taken(min)',
                    color = 'distance',
                    trendline = 'ols',
                    title = 'Relation between the Ratings and time taken')

figure.show()
```



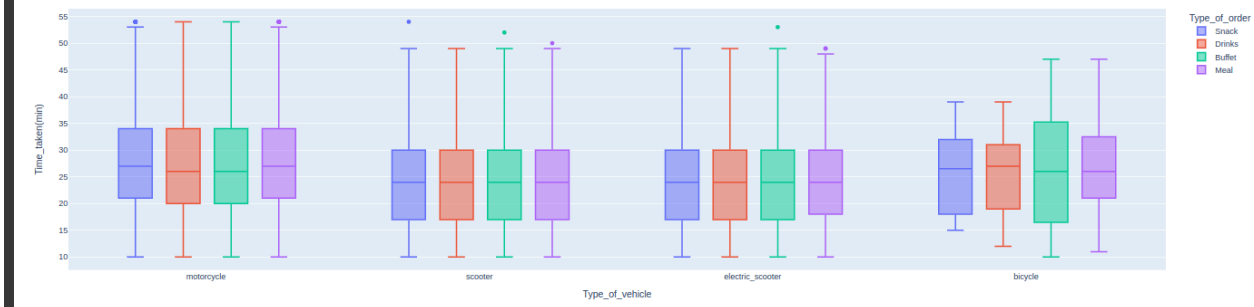
- محور x دارای برچسب "delivery_person_ratings" است.
- محور y برچسب "time_taken" است.
- عنوان نمودار به عنوان "relation between the ratings and time taken" است.

- نمودار پراکندگی دارای نقاط آبی رنگ خوشه ای است که نشان دهنده نقاط داده ای است. موقعیت هر نقطه در محور x نشان دهنده مقدار رتبه بندی آن است و موقعیت آن در محور y نشان دهنده زمان صرف شده است.
- یک مقیاس رنگ مندرج در سمت راست وجود دارد که نشان دهنده متغیر ('فاصله') است که بر شدت رنگ هر نقطه تأثیر می گذارد.

- یک خط قرمز وجود دارد که نشان دهنده یک خط روند یا رگرسیون است که رابطه کلی بین دو متغیر اصلی (رتبه بندی و زمان) را مدل می کند. شیب رو به پایین خط نشان دهنده یک رابطه منفی است، به این معنی که رتبه های بالاتر ممکن است با زمان کمتر تحویل داده شود.
یک رابطه خطی معکوس بین زمان صرف شده برای تحویل غذا و رتبه بندی تحویل دهنده وجود دارد. این بدان معناست که تحویل دهنده با رتبه‌بندی بالاتر در مقایسه با تحویل دهنده هایی با رتبه‌بندی پایین، زمان کمتری برای تحویل غذا صرف می‌کنند.

4.

```
figure = px.box(df,
                x = 'Type_of_vehicle',
                y = 'Time_taken(min)',
                color = 'Type_of_order')
figure.show()
```



نمودار جعبه ای داریم که عرض محور نوع وسیله نقلیه را مشخص می کند در حالیکه عرض محور تایم گرفته شده را نشان می دهد
چهار نوع وسیله نقلیه هست
موتور
دوچرخه
اسکوتر
اسکوتر برقی
و همچنین چهار نوع غذا برای سفارش هست که با رنگ های متفاوت نشون داده شده

با مقایسه میانه های هر جعبه متوجه میشیم که فرق چندانی تو زمان برای نوع وسیله موجود نیست بنابراین تفاوت زیادی بین زمان تحویل توسط تحویل دهنده بسته با وسیله نقلیه ای که در حال استفاده هستند و نوع غذایی که تحویل می دهند وجود ندارد.

بنابراین، ویژگی هایی که بر اساس این تحلیل بیشتر به زمان تحویل غذا کمک می کند عبارتند از:
*) سن تحویل دهنده
*) رتبه بندی تحویل دهنده
*) فاصله بین رستوران و محل تحویل

پیاده سازی مدل

```
[ ]  #splitting data
     from sklearn.model_selection import train_test_split
     # important col
     x = np.array(df[["Delivery_person_Age",
                      "Delivery_person_Ratings",
                      "distance"]])
     # label
     y = np.array(df[["Time_taken(min)"]])

     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.10,random_state = 42)
```

نحوه تقسیم داده ها به مجموعه های آموزشی و آزمایشی را با استفاده از تابع «train_test_split» از ماژول «sklearn.model_selection» نشان می دهد.

1. «from sklearn.model_selection import train_test_split»: این خط تابع «train_test_split» را از ماژول «sklearn.model_selection» فرا میخواند که برای تقسیم داده‌ها به مجموعه‌های آموزشی و آزمایشی استفاده می‌شود.

2. ` x = np.array(df[["Delivery_person_Age"، "Delivery_person_Ratings"، "distance"]] `: این خط ویژگی ها (متغیرهای ورودی) را برای مدل انتخاب می کند. یک آرایه NumPy x حاوی ستون های "Delivery_person_Age"، "Delivery_person_Ratings"، و "distance" از DataFrame df ایجاد می کند.

3. ` y = np.array(df[["Time_taken(min)]]) `: این خط متغیر هدف (برچسب) را برای مدل انتخاب می کند. یک آرایه NumPy y حاوی ستون " "Time_taken(min) از DataFrame `df` ایجاد می کند.

4. ` x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.10, random_state=42) `: این خط داده ها را به مجموعه های آموزشی و آزمایشی تقسیم می کند. 90٪ از داده ها را به آموزش ('x_train' و 'y_train') و 10٪ به آزمایش ('x_test' و 'y_test') اختصاص می دهد. پارامتر "test_size" نسبتی از مجموعه داده را برای گنجاندن در تقسیم آزمایشی مشخص می کند و "random_state" برای تولید اعداد تصادفی برای تکرارپذیری تنظیم می کند.

به طور کلی، این قطعه کد داده ها را برای آموزش یک مدل یادگیری ماشینی با تقسیم آن به ویژگی های ورودی (`x`) و برچسب های هدف (`y`) آماده می کند و سپس آن را به مجموعه های آموزشی و آزمایشی برای ارزیابی مدل تقسیم می کند.

```
# creating the LSTM neural network model
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128,return_sequences = True,input_shape = (x_train.shape[1],1)))
model.add(LSTM(64,return_sequences = False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

یک مدل شبکه عصبی LSTM (حافظه کوتاه مدت بلند مدت) را با استفاده از Keras، یک کتابخانه یادگیری عمیق محبوب ایجاد می کند. در اینجا خلاصه ای از آنچه هر قسمت انجام می دهد آورده شده است:

1. «از keras.models import Sequential» و «from keras.layers import Dense, LSTM»: این خطوط ماژول های لازم را از Keras برای ساخت مدل شبکه عصبی وارد می کنند.

2. `model = Sequential()`: این یک مدل Sequential ایجاد می کند که یک پشته خطی از لایه ها است.

3. `model.add(LSTM(128, return_sequences=True, input_shape=(x_train.shape[1], 1)))`: این یک لایه LSTM با 128 نورون به مدل اضافه می کند. «return_sequences=True» نشان می‌دهد که این لایه LSTM توالی کامل خروجی‌ها را برمی‌گرداند. `input_shape=(x_train.shape[1], 1)` شکل داده های ورودی را مشخص می کند.

4. `model.add(LSTM(64, return_sequences=False))`: این لایه LSTM دیگری با 64 نورون به مدل اضافه می کند. 'return_sequences=False' به این معنی است که این لایه LSTM تنها خروجی آخرین مرحله را برمی گرداند.

5. `model.add(Dense(25))`: این یک لایه کاملا متصل Dense با 25 واحد به مدل اضافه می کند.

6. `model.add(Dense(1))`: این یک لایه متراکم دیگر با یک واحد اضافه می کند که لایه خروجی مدل است.

7. `model.summary()`: این خط خلاصه ای از مدل را چاپ می کند و لایه ها، اشکال خروجی و تعداد پارامترهای هر لایه را نشان می دهد.

به طور کلی، این قطعه کد یک مدل شبکه عصبی با دو لایه LSTM و به دنبال آن دو لایه متراکم ایجاد می کند. مدل داده های ورودی را با شکل «(x_train.shape[1]، 1)» می گیرد و یک مقدار واحد را خروجی می دهد.

```
# training the model
model.compile(optimizer = 'adam',loss = 'mean_squared_error')
model.fit(x_train,y_train,batch_size = 1, epochs = 10)
```

```
Epoch 1/10
41033/41033 [==============================] - 283s 7ms/step - loss: 69.7029
Epoch 2/10
41033/41033 [==============================] - 280s 7ms/step - loss: 63.8293
Epoch 3/10
41033/41033 [==============================] - 307s 7ms/step - loss: 61.3168
Epoch 4/10
41033/41033 [==============================] - 272s 7ms/step - loss: 60.4354
Epoch 5/10
41033/41033 [==============================] - 271s 7ms/step - loss: 59.9861
Epoch 6/10
41033/41033 [==============================] - 284s 7ms/step - loss: 59.6107
Epoch 7/10
41033/41033 [==============================] - 281s 7ms/step - loss: 59.3948
Epoch 8/10
41033/41033 [==============================] - 284s 7ms/step - loss: 59.0068
Epoch 9/10
41033/41033 [==============================] - 285s 7ms/step - loss: 58.8234
Epoch 10/10
41033/41033 [==============================] - 290s 7ms/step - loss: 58.5813
<keras.src.callbacks.History at 0x7f93181ab700>
```

این قطعه کد نحوه آموزش مدل شبکه عصبی را با استفاده از مدل کامپایل شده و داده های آموزشی نشان می دهد. در اینجا توضیحی در مورد کد آمده است:

1. «model.compile(optimizer='adam', loss='mean_squared_error')»: این خط مدل را برای آموزش پیکربندی می کند. این بهینه ساز را به عنوان "آدام" مشخص می کند. تابع ضرر روی'mean_squared_error' تنظیم شده است، که معمولاً برای یافتن مشکلات رگرسیونی استفاده می شود که هدف آن به حداقل رساندن میانگین مجذور اختلاف بین مقادیر پیش بینی شده و واقعی است.

2. ` model.fit(x_train, y_train, batch_size=1, epochs=10)`: این خط مدل را با داده های آموزشی مطابقت می دهد. «x_train» و «y_train» ویژگی‌های ورودی و برچسب‌های هدف برای آموزش هستند. "batch_size=1" تعداد نمونه ها را در هر به روز رسانی گرادیان مشخص می کند. در این حالت، مدل وزن‌ها را پس از هر نمونه به‌روزرسانی می‌کند. پارامتر 'epochs=10' تعداد دفعاتی را که مدل در کل مجموعه داده آموزشی تکرار می شود را مشخص می کند.

به طور کلی، این قطعه کد مدل را با بهینه ساز و تابع ضرر مشخص شده کامپایل می کند و سپس مدل را بر روی داده های آموزشی برای 10 دوره با اندازه دسته های 1 آموزش می دهد. بین مقادیر پیش بینی شده و واقعی

```
print("Fodd Delivery Time Prediction")
a = int(input("Age of Delivery Partner: "))
b = float(input("Ratings of Previous Delivery: "))
c = int(input("Total Distance: "))

features = np.array([[a,b,c]])
print("Predicted delivery Time in Minutes : ",model.predict(features))
```

```
Fodd Delivery Time Prediction
Age of Delivery Partner: 44
Ratings of Previous Delivery: 3
Total Distance: 24
1/1 [==============================] - 1s 788ms/step
Predicted delivery Time in Minutes :  [[35.417587]]
```

این قطعه کد یک اسکریپت تعاملی ساده برای پیش بینی با استفاده از مدل آموزش دیده است. در اینجا توضیحی در مورد کد آمده است:

1. print("Food Delivery Time Prediction"): این خط پیش‌بینی را چاپ می‌کند، که نشان می‌دهد ورودی‌های زیر برای پیش‌بینی زمان تحویل استفاده خواهند شد.

2. `a = int(input("Age of Delivery Partner:"))`: این خط از کاربر می خواهد که سن تحویل دهنده را وارد کند. تابع 'input' ورودی کاربر را به عنوان یک رشته می گیرد و 'int' برای تبدیل آن به یک عدد صحیح استفاده می شود.

3. `b = float(input("Ratings of Previous Delivery:"))`: این خط از کاربر می خواهد تا رتبه بندی تحویل دهنده را وارد کند. تابع "input" ورودی کاربر را به عنوان یک رشته می گیرد و از "float" برای تبدیل آن به یک عدد ممیز شناور استفاده می شود.

4. `c = int(input("Total Distance: "))`: این خط از کاربر می خواهد که فاصله کل تحویل را وارد کند. تابع 'input' ورودی کاربر را به عنوان یک رشته می گیرد و 'int' برای تبدیل آن به یک عدد صحیح استفاده می شود.

5. features = np.array([[a, b, c]]): این خط یک "features" آرایه NumPy ایجاد می کند که حاوی مقادیر ارائه شده توسط کاربر برای سن شریک تحویل، رتبه بندی های قبلی و فاصله کلی است. این آرایه همانند داده های آموزشی قالب بندی شده است.

6. `print("Predicted delivery Time in Minutes : ", model.predict(features))`: این خط از مدل آموزش دیده برای پیش بینی بر اساس ورودی کاربر استفاده می کند. با فراخوانی روش «پیش‌بینی» مدل و ارسال ورودی کاربر به عنوان ویژگی، زمان تحویل پیش‌بینی‌شده را در چند دقیقه چاپ می‌کند.

به طور کلی، این قطعه کد یک رابط ساده برای کاربر ایجاد می کند تا اطلاعات تحویل دهنده را وارد کند و سپس از مدل آموزش دیده برای پیش بینی زمان تحویل بر اساس ورودی های ارائه شده استفاده می کند.

حالا چیزی که من به کد اضافه کردم:

```python
from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the model on the test set
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2:.2f}')
```

```
143/143 [==============================] - 1s 8ms/step
Mean Squared Error: 58.40
R-squared: 0.33
```

این قطعه کد نحوه ارزیابی عملکرد یک مدل یادگیری ماشین را با استفاده از معیارهایی مانند میانگین مربعات خطا (MSE) و R-squared (R2) نشان می‌دهد. در اینجا توضیحی در مورد کد آمده است:

1. «from sklearn.metrics import mean_squared_error, r2_score»: این خط توابع ضروری «mean_squared_error» و «r2_score» را از ماژول «sklearn.metrics» وارد می کند. این توابع معمولاً برای ارزیابی عملکرد مدل‌های رگرسیون استفاده می‌شوند.

2. `y_pred = model.predict(x_test)`: این خط از مدل آموزش دیده برای پیش بینی مجموعه تست (`x_test`) استفاده می کند و مقادیر پیش بینی شده را در متغیر `y_pred` ذخیره می کند.

3. `mse = mean_squared_error(y_test, y_pred)`: این خط میانگین مربعات خطا (MSE) را بین مقادیر هدف واقعی (`y_test`) و مقادیر پیش بینی شده (`y_pred`) محاسبه می کند. MSE میانگین اختلاف مجذور بین مقادیر پیش بینی شده و واقعی را اندازه گیری می کند.

4. `r2 = r2_score(y_test, y_pred)`: این خط امتیاز R-squared (R2) را محاسبه می کند که یک معیار آماری است که نشان دهنده نسبت واریانس در متغیر وابسته است که از روی متغیرهای مستقل قابل پیش بینی است. این نشان می دهد که مدل چقدر با داده ها در مقایسه با یک مدل پایه مطابقت دارد.

5. `print(f'Mean Squared Error: {mse:.2f}')`: این خط مقدار میانگین مربعات خطا (MSE) را با دو رقم اعشار چاپ می کند. مقادیر کمتر MSE نشان دهنده عملکرد بهتر مدل از نظر دقت پیش بینی است.

6. `print(f'R-squared: {r2:.2f}')`: این خط امتیاز R-squared (R2) را با دو رقم اعشار چاپ می کند. مقادیر R2 از 0 تا 1 متغیر است، جایی که 1 نشان دهنده تناسب کامل و 0 نشان می دهد که مدل هیچ یک از واریانس ها در متغیر هدف را توضیح نمی دهد.

به طور کلی، این قطعه کد عملکرد مدل آموزش دیده را در مجموعه آزمایشی با محاسبه و نمایش معیارهای میانگین مربعات خطا (MSE) و R-squared (R2) ارزیابی می کند. این معیارها بینش هایی را در مورد دقت مدل و چگونگی توضیح واریانس در متغیر هدف ارائه می دهند.

```python
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(x_train, y_train)
rf_pred = rf_model.predict(x_test)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)
print(f'Random Forest MSE: {rf_mse:.2f}')
print(f'Random Forest R-squared: {rf_r2:.2f}')

# XGBoost Regressor
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(x_train, y_train)
xgb_pred = xgb_model.predict(x_test)
xgb_mse = mean_squared_error(y_test, xgb_pred)
xgb_r2 = r2_score(y_test, xgb_pred)
print(f'XGBoost MSE: {xgb_mse:.2f}')
print(f'XGBoost R-squared: {xgb_r2:.2f}')
```

```
<ipython-input-20-f160b380b571>:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

Random Forest MSE: 64.88
Random Forest R-squared: 0.26
XGBoost MSE: 54.89
XGBoost R-squared: 0.37
```

این قطعه کد نحوه آموزش و ارزیابی دو مدل رگرسیون مختلف Random Forest Regressor و XGBoost Regressor را با استفاده از میانگین مربعات خطا (MSE) و R-squared (R2) نشان می دهد.

1. «from sklearn.ensemble import RandomForestRegressor»: این خط کلاس RandomForestRegressor را از ماژول sklearn.ensemble وارد می کند. RandomForest یک روش یادگیری گروهی است که چندین درخت تصمیم می سازد و پیش بینی های آنها را برای بهبود تعمیم پذیری و استحکام ترکیب می کند.

2. «from xgboost import XGBRegressor»: این خط کلاس XGBRegressor را از کتابخانه xgboost وارد می‌کند. XGBoost یک کتابخانه بهینه سازی شده برای تقویت گرادیان است که به دلیل سرعت و عملکرد در مسابقات یادگیری ماشینی شناخته شده است.

3. «from sklearn.metrics import mean_squared_error, r2_score»: این خط توابع mean_squared_error و r2_score را از ماژول sklearn.metrics وارد می کند. این توابع معمولا برای ارزیابی مدل های رگرسیون بر اساس عملکرد پیش بینی آنها استفاده می شود.

4. `rf_model = RandomForestRegressor(n_estimators=100, random_state=42)`: این خط نمونه ای از کلاس RandomForestRegressor با 100 برآوردگر (درخت تصمیم) و حالت تصادفی 42 ایجاد می کند. حالت تصادفی برای تکرارپذیری نتایج تنظیم شده است.

5. `rf_model.fit(x_train, y_train)`: این خط با مدل جنگل تصادفی (`rf_model`) در داده های آموزشی (`x_train` و `y_train`) مطابقت دارد. مدل رابطه بین ویژگی های ورودی (`x_train`) و متغیر هدف (`y_train`) را یاد می گیرد.

6. `rf_pred = rf_model.predict(x_test)`: این خط از مدل جنگل تصادفی آموزش دیده برای پیش بینی داده های آزمون (`x_test`) استفاده می کند و مقادیر پیش بینی شده را در متغیر `rf_pred` ذخیره می کند.

7. `rf_mse = mean_squared_error(y_test, rf_pred)`: این خط میانگین مربعات خطا (MSE) را بین مقادیر هدف واقعی (`y_test`) و مقادیر پیش بینی شده از مدل جنگل تصادفی (`rf_pred`) محاسبه می کند.

8. `rf_r2 = r2_score(y_test, rf_pred)`: این خط امتیاز R-squared (R2) را برای ارزیابی عملکرد مدل Random Forest بر روی داده های آزمون محاسبه می کند. R2 اندازه گیری می کند که مدل چقدر با داده ها در مقایسه با یک مدل پایه مطابقت دارد.

9. «print(f'Random Forest R-squared: {rf_r2:.2f}')»:این خط امتیاز مربع R (R2) مدل جنگل تصادفی را با دو رقم اعشار چاپ می کند، که نشان می دهد مدل چقدر واریانس در متغیر هدف را توضیح می دهد.

به طور کلی، این قطعه کد یک مدل Random Forest Regressor را آموزش می دهد، عملکرد آن را با استفاده از معیارهای MSE و R2 ارزیابی می کند و امتیاز R2 مدل Random Forest را چاپ می کند.