Jatin Soneta
Batch B2
LY IT 36

AIDS-II
Assignment No : 2

Aim : Develop cognitive application for Logistics using Neural Network.

Software Required : Google Collab.

Code :

Porter Delivery Time Estimation

Importing libraries

```
[ ] import tensorflow as tf
    print(tf.__version__)
```

```
2.17.0
```

```
[ ] #for reading and handling the data
    import pandas as pd
    import numpy as np
    import os

    #for visualizinng and analyzing it
    import matplotlib.pyplot as plt
    import seaborn as sns

    #data preprocessing
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split

    #random forest model training
    from sklearn.metrics import mean_squared_error
    from sklearn.metrics import r2_score
    from sklearn.metrics import mean_absolute_error
    from sklearn.ensemble import RandomForestRegressor
```

```python
#ann training
from tensorflow.keras import Model
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense,Dropout,BatchNormalization,LeakyReLU
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.losses import MeanAbsolutePercentageError

from tensorflow.keras.metrics import MeanAbsolutePercentageError, RootMeanSquaredError, MeanAbsoluteError
from tensorflow.keras.optimizers import SGD, Adam
```

```python
sns.set(rc={'figure.figsize':(11.7,8.27)})
```

Loading the data from kaggle

```python
df=pd.read_csv('/content/porter_data.csv')
```

Printing the head and information of the data to get an understanding of it

```python
df.head()
```

	market_id	created_at	actual_delivery_time	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dasher
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	1.0	4	3441	4	557	1239	33.
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	46	2.0	1	1900	1	1400	1400	1.
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	36	3.0	4	4771	3	820	1604	8.
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	38	1.0	1	1525	1	1525	1525	5.
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	38	1.0	2	3620	2	1425	2195	5.

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   market_id                  175777 non-null  float64
 1   created_at                 175777 non-null  object
 2   actual_delivery_time       175777 non-null  object
 3   store_primary_category     175777 non-null  int64
 4   order_protocol             175777 non-null  float64
 5   total_items                175777 non-null  int64
 6   subtotal                   175777 non-null  int64
 7   num_distinct_items         175777 non-null  int64
 8   min_item_price             175777 non-null  int64
 9   max_item_price             175777 non-null  int64
 10  total_onshift_dashers      175777 non-null  float64
 11  total_busy_dashers         175777 non-null  float64
 12  total_outstanding_orders   175777 non-null  float64
```

Data preprocessing

```
[ ] df['created_at']=pd.to_datetime(df['created_at'])
    df['actual_delivery_time']=pd.to_datetime(df['actual_delivery_time'])
```

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     175777 non-null  float64
 1   created_at                                    175777 non-null  datetime64[ns]
 2   actual_delivery_time                          175777 non-null  datetime64[ns]
 3   store_primary_category                        175777 non-null  int64
 4   order_protocol                                175777 non-null  float64
 5   total_items                                   175777 non-null  int64
 6   subtotal                                      175777 non-null  int64
 7   num_distinct_items                            175777 non-null  int64
 8   min_item_price                                175777 non-null  int64
 9   max_item_price                                175777 non-null  int64
 10  total_onshift_dashers                         175777 non-null  float64
 11  total_busy_dashers                            175777 non-null  float64
 12  total_outstanding_orders                      175777 non-null  float64
 13  estimated_store_to_consumer_driving_duration  175777 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(6)
memory usage: 18.8 MB
```

```
[ ] df['time_taken']=df['actual_delivery_time'] - df['created_at']
```

```
[ ] df.head()
```

	market_id	created_at	actual_delivery_time	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dashers
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	1.0	4	3441	4	557	1239	33.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	46	2.0	1	1900	1	1400	1400	1.0
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	36	3.0	4	4771	3	820	1604	8.0
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	38	1.0	1	1525	1	1525	1525	5.0
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	38	1.0	2	3620	2	1425	2195	5.0

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 15 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     175777 non-null  float64
 1   created_at                                    175777 non-null  datetime64[ns]
 2   actual_delivery_time                          175777 non-null  datetime64[ns]
 3   store_primary_category                        175777 non-null  int64
 4   order_protocol                                175777 non-null  float64
 5   total_items                                   175777 non-null  int64
 6   subtotal                                      175777 non-null  int64
 7   num_distinct_items                            175777 non-null  int64
 8   min_item_price                                175777 non-null  int64
 9   max_item_price                                175777 non-null  int64
 10  total_onshift_dashers                         175777 non-null  float64
 11  total_busy_dashers                            175777 non-null  float64
 12  total_outstanding_orders                      175777 non-null  float64
 13  estimated_store_to_consumer_driving_duration  175777 non-null  float64
```

```python
df['time_taken_mins']=pd.to_timedelta(df['time_taken'])/pd.Timedelta('60s')
```

```python
df.head()
```

	market_id	created_at	actual_delivery_time	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dashers
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	1.0	4	3441	4	557	1239	33.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	46	2.0	1	1900	1	1400	1400	1.0
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	36	3.0	4	4771	3	820	1604	8.0
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	38	1.0	1	1525	1	1525	1525	5.0
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	38	1.0	2	3620	2	1425	2195	5.0

```python
df['hour']=df['created_at'].dt.hour
df['day']=df['created_at'].dt.dayofweek
```

```python
df.head()
```

	market_id	created_at	actual_delivery_time	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dashers
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	1.0	4	3441	4	557	1239	33.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	46	2.0	1	1900	1	1400	1400	1.0
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	36	3.0	4	4771	3	820	1604	8.0
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	38	1.0	1	1525	1	1525	1525	5.0
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	38	1.0	2	3620	2	1425	2195	5.0

```python
df.drop(['time_taken','created_at','actual_delivery_time'],axis=1,inplace=True)
```

Checking null values in the data

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 15 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     175777 non-null  float64
 1   store_primary_category                        175777 non-null  int64
 2   order_protocol                                175777 non-null  float64
 3   total_items                                   175777 non-null  int64
 4   subtotal                                      175777 non-null  int64
 5   num_distinct_items                            175777 non-null  int64
 6   min_item_price                                175777 non-null  int64
 7   max_item_price                                175777 non-null  int64
 8   total_onshift_dashers                         175777 non-null  float64
 9   total_busy_dashers                            175777 non-null  float64
 10  total_outstanding_orders                      175777 non-null  float64
 11  estimated_store_to_consumer_driving_duration  175777 non-null  float64
 12  time_taken_mins                               175777 non-null  float64
 13  hour                                          175777 non-null  int32
 14  day                                           175777 non-null  int32
dtypes: float64(7), int32(2), int64(6)
memory usage: 18.8 MB
```

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

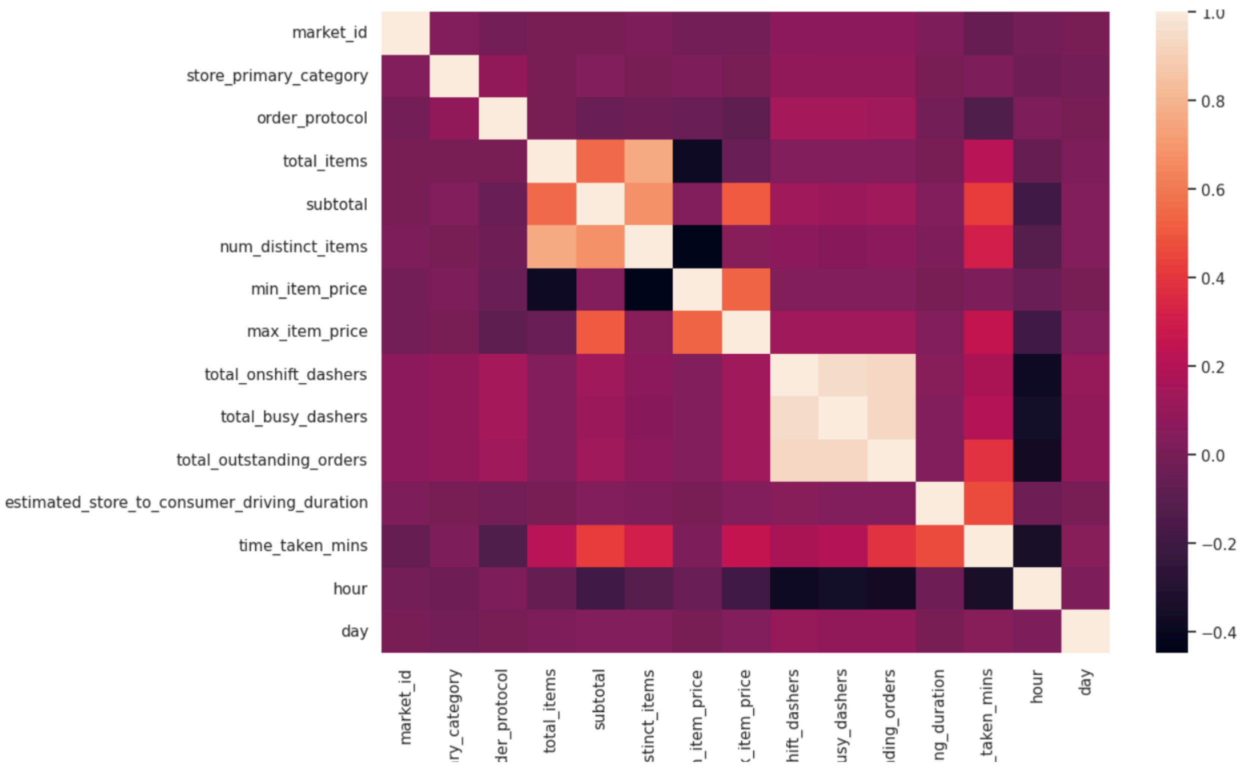
market_id	0
store_primary_category	0
order_protocol	0
total_items	0
subtotal	0
num_distinct_items	0
min_item_price	0
max_item_price	0
total_onshift_dashers	0
total_busy_dashers	0
total_outstanding_orders	0
estimated_store_to_consumer_driving_duration	0
time_taken_mins	0
hour	0
day	0

dtype: int64

dropping null values from the data(if present)

Plotting correlation to get an idea of the data

```
[ ] sns.heatmap(df.corr())
```



we have one categorical column which we will change to integer for model

```
[ ] df['store_primary_category']=df['store_primary_category'].astype('category').cat.codes
```

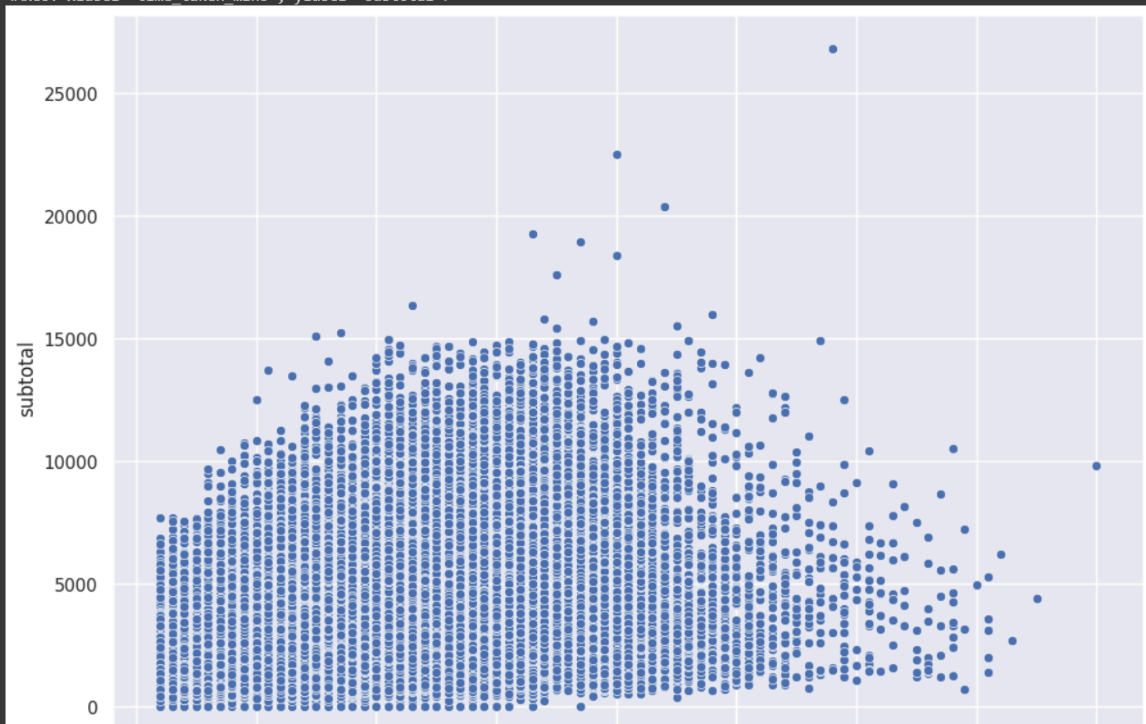
```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 15 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     175777 non-null  float64
 1   store_primary_category                        175777 non-null  int8
 2   order_protocol                                175777 non-null  float64
 3   total_items                                   175777 non-null  int64
 4   subtotal                                      175777 non-null  int64
 5   num_distinct_items                            175777 non-null  int64
 6   min_item_price                                175777 non-null  int64
 7   max_item_price                                175777 non-null  int64
 8   total_onshift_dashers                         175777 non-null  float64
 9   total_busy_dashers                            175777 non-null  float64
 10  total_outstanding_orders                      175777 non-null  float64
 11  estimated_store_to_consumer_driving_duration  175777 non-null  float64
 12  time_taken_mins                               175777 non-null  float64
 13  hour                                          175777 non-null  int32
 14  day                                           175777 non-null  int32
dtypes: float64(7), int32(2), int64(5), int8(1)
memory usage: 17.6 MB
```

Data Visualization and Cleaning

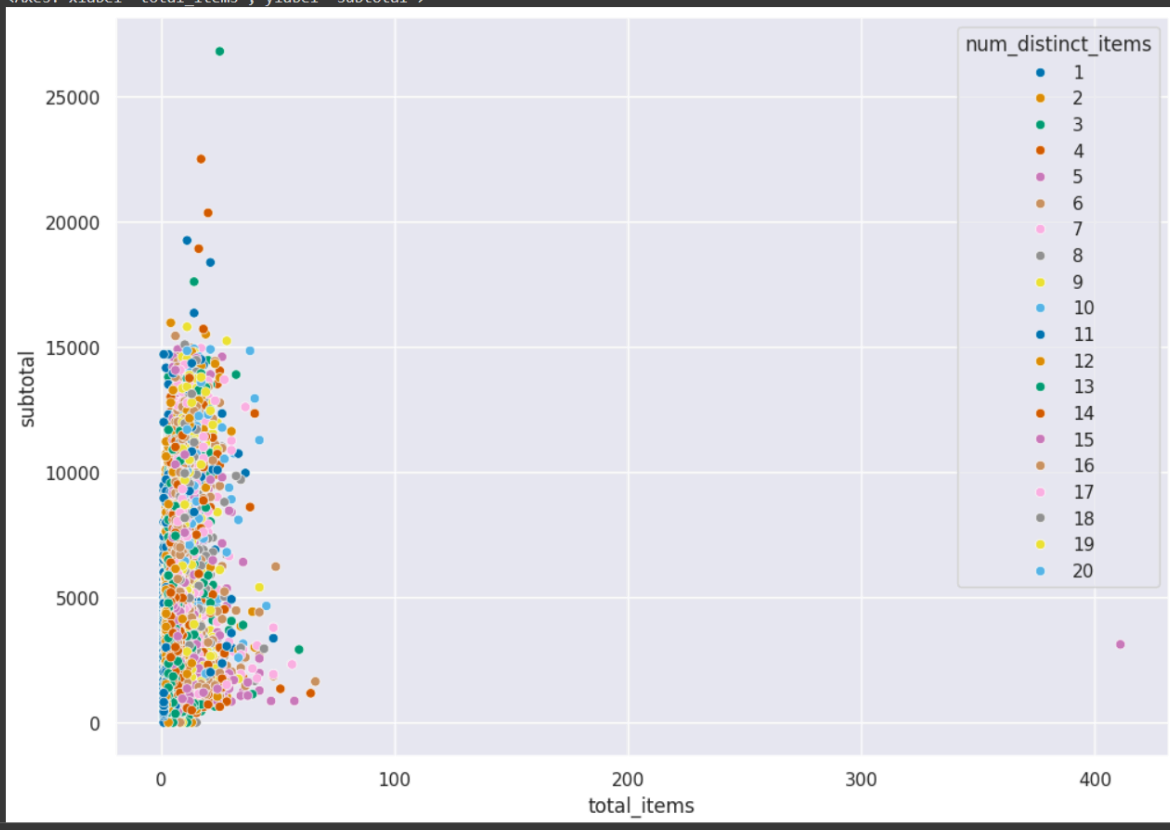
```
sns.scatterplot(x='time_taken_mins',y='subtotal',data=df)
```

<Axes: xlabel='time_taken_mins', ylabel='subtotal'>



```
sns.scatterplot(x='total_items',y='subtotal',hue='num_distinct_items',palette='colorblind',data=df)
```

```
<Axes: xlabel='total_items', ylabel='subtotal'>
```



```
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
model1=LocalOutlierFactor()
#model1.fit(df)
df['lof_anomaly_score']=model1.fit_predict(df)
```

```
[ ] print("number of outliers : ",(len(df.loc[(df['lof_anomaly_score'] == -1)])))
    df=df.loc[(df['lof_anomaly_score'] == 1)]
```

```
number of outliers :  831
```

```
[ ] df.drop(['lof_anomaly_score'],axis=1,inplace=True)
```

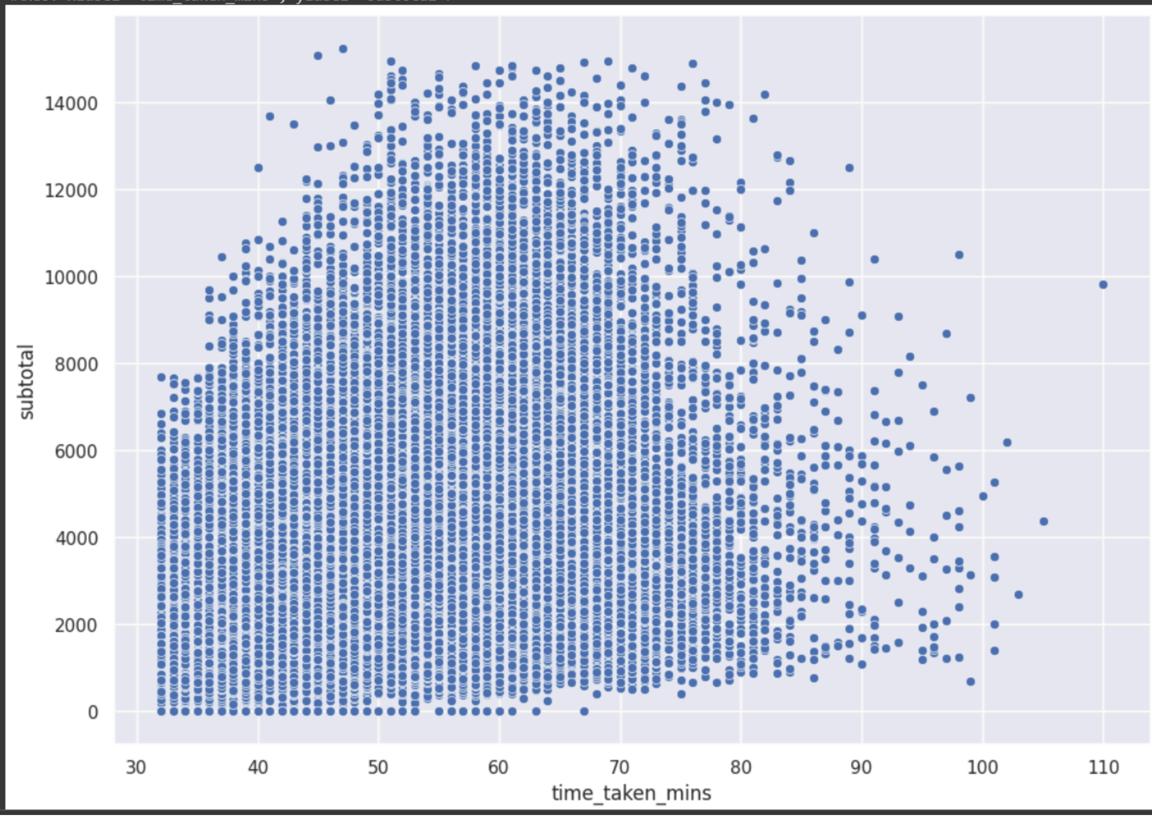
```
<ipython-input-27-a629ff9cb1da>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df.drop(['lof_anomaly_score'],axis=1,inplace=True)
```

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 174946 entries, 0 to 175776
Data columns (total 15 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     174946 non-null  float64
 1   store_primary_category                        174946 non-null  int8
 2   order_protocol                                174946 non-null  float64
 3   total_items                                   174946 non-null  int64
 4   subtotal                                      174946 non-null  int64
 5   num_distinct_items                            174946 non-null  int64
 6   min_item_price                                174946 non-null  int64
 7   max_item_price                                174946 non-null  int64
 8   total_onshift_dashers                         174946 non-null  float64
 9   total_busy_dashers                            174946 non-null  float64
 10  total_outstanding_orders                      174946 non-null  float64
 11  estimated_store_to_consumer_driving_duration  174946 non-null  float64
 12  time_taken_mins                               174946 non-null  float64
 13  hour                                          174946 non-null  int32
```

```
sns.scatterplot(x='time_taken_mins',y='subtotal',data=df)
```

```
<Axes: xlabel='time_taken_mins', ylabel='subtotal'>
```

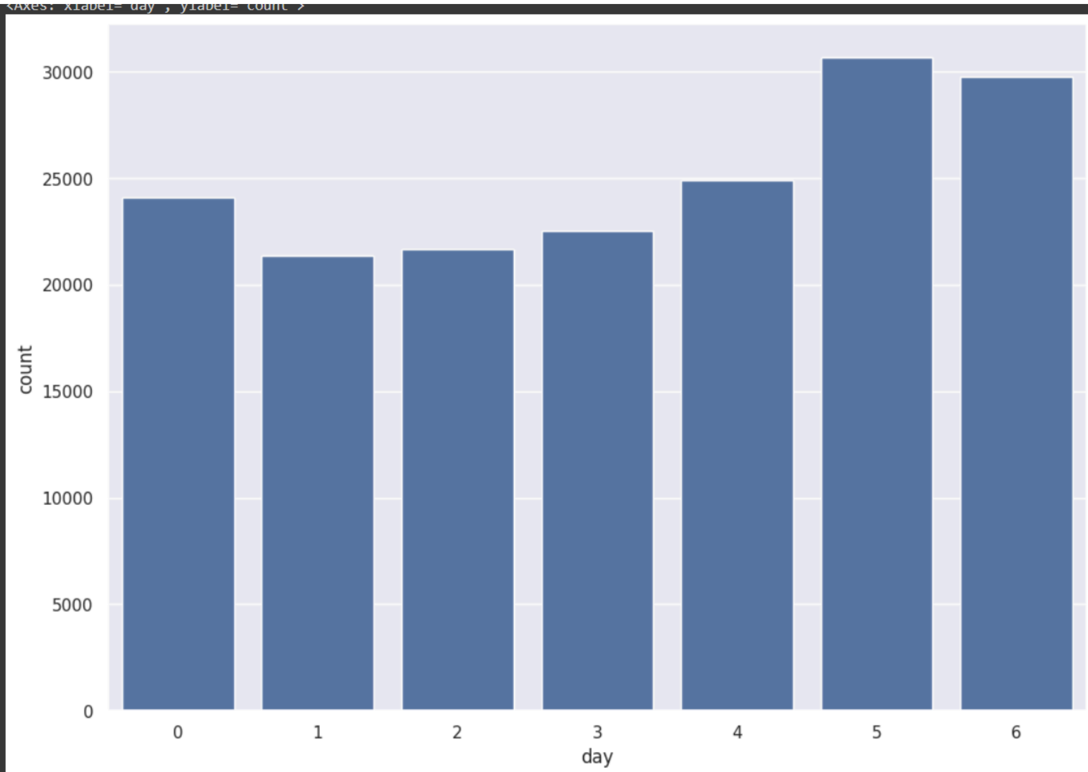



```
df.columns
```

```
Index(['market_id', 'store_primary_category', 'order_protocol', 'total_items',
       'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_dashers', 'total_busy_dashers',
       'total_outstanding_orders',
       'estimated_store_to_consumer_driving_duration', 'time_taken_mins',
       'hour', 'day'],
      dtype='object')
```

```
sns.countplot(x=df.day)
```

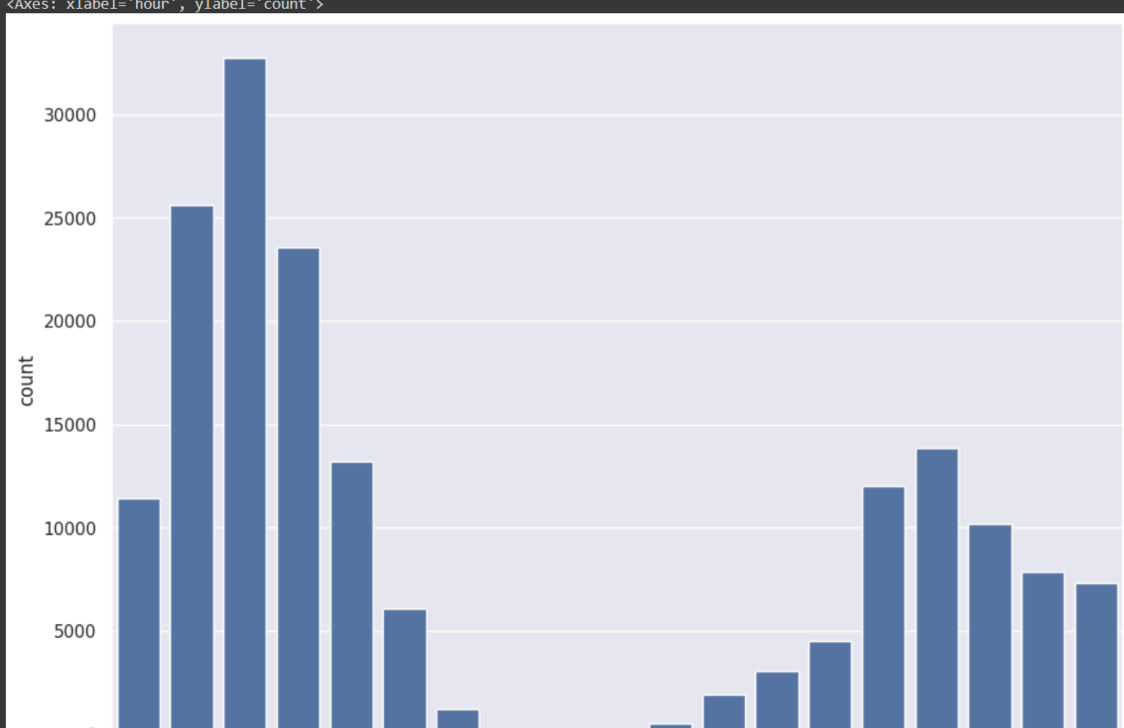
```
<Axes: xlabel='day', ylabel='count'>
```



a little more orders on the weekends

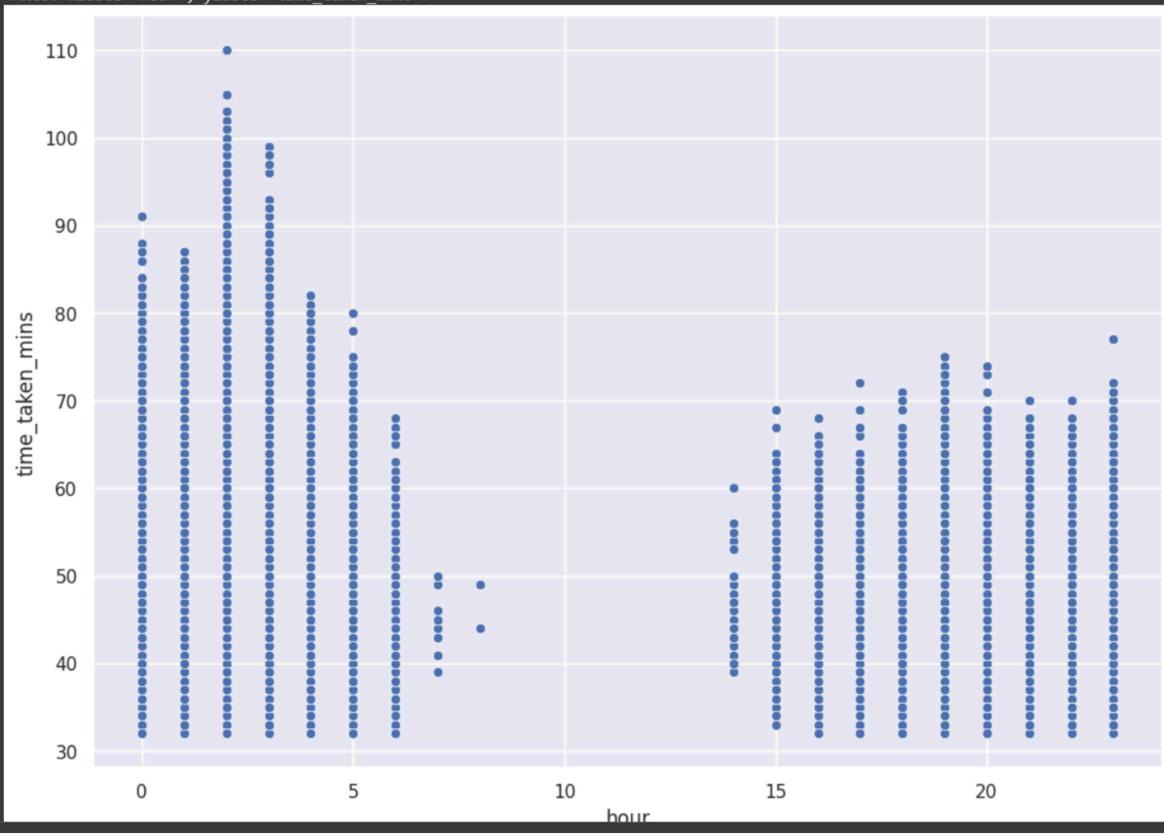
```
sns.countplot(x=df.hour)
```

```
<Axes: xlabel='hour', ylabel='count'>
```



```
sns.scatterplot(x='hour',y='time_taken_mins',data=df)
```

```
<Axes: xlabel='hour', ylabel='time_taken_mins'>
```



Data Splitting and Modelling

```
y=df['time_taken_mins']
x=df.drop(['time_taken_mins'],axis=1)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x.head()
```

	market_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dashers	total_busy_dashers	total_outstar
0	1.0	4	1.0	4	3441	4	557	1239	33.0	14.0	
1	2.0	46	2.0	1	1900	1	1400	1400	1.0	2.0	
2	2.0	36	3.0	4	4771	3	820	1604	8.0	6.0	
3	1.0	38	1.0	1	1525	1	1525	1525	5.0	6.0	
4	1.0	38	1.0	2	3620	2	1425	2195	5.0	5.0	

Neural Networks

Scalling the data to feed before neural network

```
from sklearn import preprocessing
scaler=preprocessing.MinMaxScaler()
x_scaled=scaler.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=42)
```

+ Code + Text

```
model=Sequential()
model.add(Dense(14,kernel_initializer='normal',activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(1024,activation='relu'))
model.add(Dense(256,activation='relu'))
model.add(Dense(1,activation='linear'))
```

```
from tensorflow.keras.optimizers import Adam
adam=Adam(learning_rate=0.01)
model.compile(loss='mse',optimizer=adam,metrics=['mse','mae'])
history=model.fit(X_train,y_train,epochs=30,batch_size=512,verbose=1,validation_split=0.2)
```

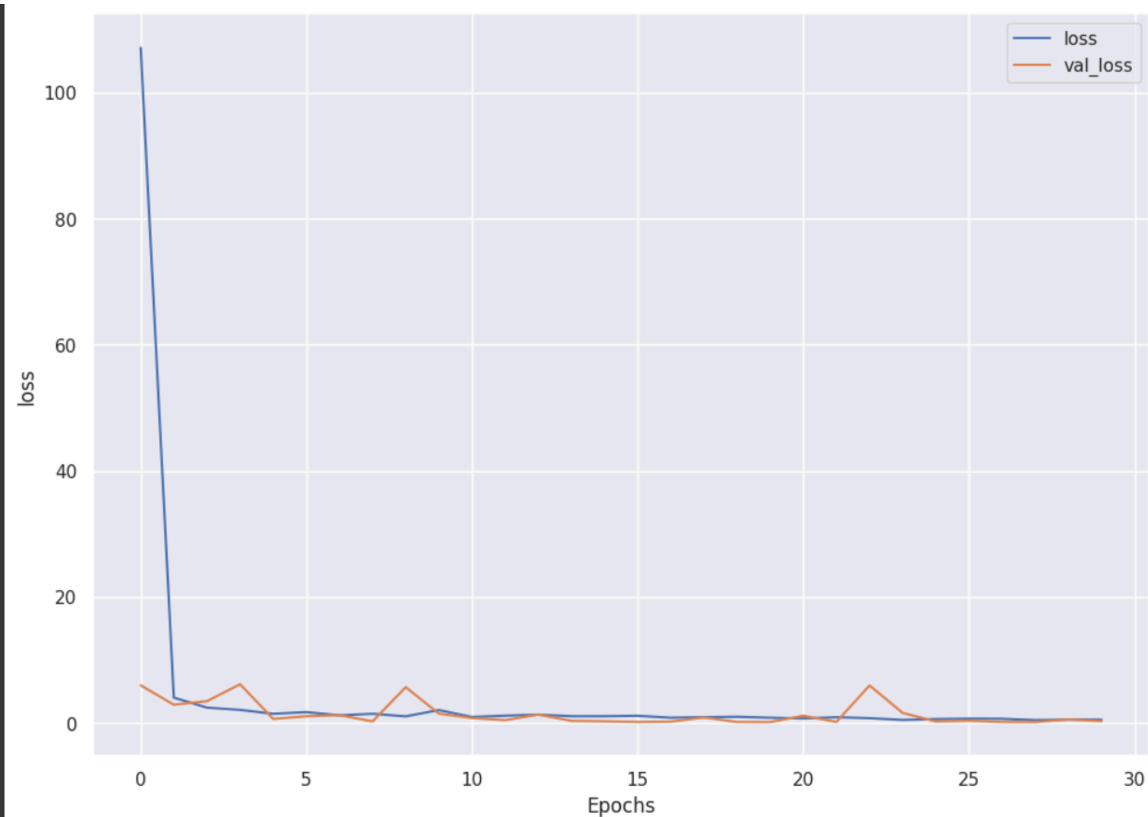
```
Epoch 2/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 22s 75ms/step - loss: 4.6148 - mae: 1.5679 - mse: 4.6148 - val_loss: 2.8865 - val_mae: 1.2347 - val_mse: 2.8865
Epoch 3/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 16s 72ms/step - loss: 2.6407 - mae: 1.2134 - mse: 2.6407 - val_loss: 3.4450 - val_mae: 1.5465 - val_mse: 3.4450
Epoch 4/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 16s 74ms/step - loss: 2.0042 - mae: 1.0908 - mse: 2.0042 - val_loss: 6.1359 - val_mae: 2.2703 - val_mse: 6.1359
Epoch 5/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 66ms/step - loss: 1.5531 - mae: 0.9583 - mse: 1.5531 - val_loss: 0.6342 - val_mae: 0.6247 - val_mse: 0.6342
Epoch 6/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 28s 101ms/step - loss: 1.5289 - mae: 0.9719 - mse: 1.5289 - val_loss: 1.0595 - val_mae: 0.8614 - val_mse: 1.0595
Epoch 7/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 33s 67ms/step - loss: 1.0348 - mae: 0.7912 - mse: 1.0348 - val_loss: 1.2336 - val_mae: 0.9259 - val_mse: 1.2336
Epoch 8/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 69ms/step - loss: 1.4041 - mae: 0.9525 - mse: 1.4041 - val_loss: 0.2395 - val_mae: 0.3782 - val_mse: 0.2395
Epoch 9/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 17s 76ms/step - loss: 0.9676 - mae: 0.7701 - mse: 0.9676 - val_loss: 5.6830 - val_mae: 2.3091 - val_mse: 5.6830
```


```
Epoch 6/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 28s 101ms/step - loss: 1.5289 - mae: 0.9719 - mse: 1.5289 - val_loss: 1.0595 - val_mae: 0.8614 - val_mse: 1.0595
Epoch 7/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 33s 67ms/step - loss: 1.0348 - mae: 0.7912 - mse: 1.0348 - val_loss: 1.2336 - val_mae: 0.9259 - val_mse: 1.2336
Epoch 8/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 69ms/step - loss: 1.4041 - mae: 0.9525 - mse: 1.4041 - val_loss: 0.2395 - val_mae: 0.3782 - val_mse: 0.2395
Epoch 9/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 17s 76ms/step - loss: 0.9676 - mae: 0.7701 - mse: 0.9676 - val_loss: 5.6830 - val_mae: 2.3091 - val_mse: 5.6830
```


```
Epoch 10/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 16s 73ms/step - loss: 3.0217 - mae: 1.2476 - mse: 3.0217 - val_loss: 1.4647 - val_mae: 1.0925 - val_mse: 1.4647
Epoch 11/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 70ms/step - loss: 0.7654 - mae: 0.6903 - mse: 0.7654 - val_loss: 0.7743 - val_mae: 0.7436 - val_mse: 0.7743
Epoch 12/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 66ms/step - loss: 1.0219 - mae: 0.7639 - mse: 1.0219 - val_loss: 0.4415 - val_mae: 0.5556 - val_mse: 0.4415
Epoch 13/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 21s 67ms/step - loss: 1.2893 - mae: 0.8697 - mse: 1.2893 - val_loss: 1.3270 - val_mae: 1.0624 - val_mse: 1.3270
Epoch 14/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 1.1587 - mae: 0.8570 - mse: 1.1587 - val_loss: 0.3210 - val_mae: 0.4556 - val_mse: 0.3210
Epoch 15/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 21s 67ms/step - loss: 1.0512 - mae: 0.8216 - mse: 1.0512 - val_loss: 0.2555 - val_mae: 0.4122 - val_mse: 0.2555
Epoch 16/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 21s 69ms/step - loss: 1.2820 - mae: 0.8842 - mse: 1.2820 - val_loss: 0.1476 - val_mae: 0.3093 - val_mse: 0.1476
Epoch 17/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 20s 67ms/step - loss: 0.7086 - mae: 0.6464 - mse: 0.7086 - val_loss: 0.2041 - val_mae: 0.3633 - val_mse: 0.2041
Epoch 18/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 1.1399 - mae: 0.8188 - mse: 1.1399 - val_loss: 0.8746 - val_mae: 0.8580 - val_mse: 0.8746
Epoch 19/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 68ms/step - loss: 1.1643 - mae: 0.8541 - mse: 1.1643 - val_loss: 0.1443 - val_mae: 0.3085 - val_mse: 0.1443
Epoch 20/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 21s 68ms/step - loss: 0.5999 - mae: 0.6039 - mse: 0.5999 - val_loss: 0.1399 - val_mae: 0.3036 - val_mse: 0.1399
Epoch 21/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 0.6872 - mae: 0.6777 - mse: 0.6872 - val_loss: 1.1405 - val_mae: 0.9899 - val_mse: 1.1405
Epoch 22/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 21s 68ms/step - loss: 0.8494 - mae: 0.7199 - mse: 0.8494 - val_loss: 0.1626 - val_mae: 0.3257 - val_mse: 0.1626
Epoch 23/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 22s 74ms/step - loss: 0.4172 - mae: 0.4960 - mse: 0.4172 - val_loss: 5.9506 - val_mae: 2.3522 - val_mse: 5.9506
Epoch 24/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 0.9181 - mae: 0.6901 - mse: 0.9181 - val_loss: 1.5831 - val_mae: 1.1874 - val_mse: 1.5831
Epoch 25/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 0.6158 - mae: 0.5809 - mse: 0.6158 - val_loss: 0.2253 - val_mae: 0.3792 - val_mse: 0.2253
Epoch 26/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - loss: 0.7046 - mae: 0.6477 - mse: 0.7046 - val_loss: 0.3451 - val_mae: 0.4970 - val_mse: 0.3451
Epoch 27/30
219/219 ━━━━━━━━━━━━━━━━━━━━ 20s 66ms/step - loss: 0.4009 - mae: 0.5037 - mse: 0.4009 - val_loss: 0.1429 - val_mae: 0.3081 - val_mse: 0.1429
```

we plot train and validation loss throughout training

```
def plot_history(history,key):
    plt.plot(history.history[key])
    plt.plot(history.history['val_'+key])
    plt.xlabel("Epochs")
    plt.ylabel(key)
    plt.legend([key,'val_'+key])
    plt.show()
#plot the history
plot_history(history,'loss')
```



val loss is below training loss so our model is not overfitting

```
z= model.predict(X_test)
```

```
1094/1094 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step
```

```
r2_score(y_test, z)
```

```
0.9968542686597435
```

```
mse = mean_squared_error(y_test, z)
rmse = mse**.5
print("mse : ",mse)
print("rmse : ",rmse)
print("errors for neural net")
mae = mean_absolute_error(y_test, z)
print("mae : ",mae)
```

```
mse :  0.2706599711738081
rmse :  0.5202499122285443
errors for neural net
mae :  0.4301021189442973
```

```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, z)
```

```
0.009434215653366511
```