```
!pip install PyDrive
```

```
Requirement already satisfied: PyDrive in /usr/local/lib/python3.7/dist-packages (1.3
Requirement already satisfied: PyYAML>=3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: google-api-python-client>=1.2 in /usr/local/lib/python
Requirement already satisfied: oauth2client>=4.0.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3
Requirement already satisfied: google-api-core<2dev,>=1.21.0 in /usr/local/lib/python
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: google-auth>=1.16.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: six<2dev,>=1.13.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: httplib2<1dev,>=0.15.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from g
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.7
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
```

```
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials


auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)


import zipfile
from google.colab import drive

zip_ref = zipfile.ZipFile("/content/drive/MyDrive/data.zip", 'r') #import zip file
zip_ref.extractall("/tmp") #extract zip file
zip_ref.close()


!ls /tmp/data
```

```
drivers.csv  pings.csv  test.csv
```

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns

#load all 3 datasets
driver_df =  pd.read_csv("/tmp/data/drivers.csv")
ping_df = pd.read_csv("/tmp/data/pings.csv")
test_df = pd.read_csv("/tmp/data/test.csv")

print("shape of the driver_df:",driver_df.shape)
print("shape of the ping_df:",ping_df.shape)
print("shape of the test_df:",test_df.shape)

print("********* Head of the driver_df**************")
print(driver_df.head())
print("********* Head of the ping_df**************")
print(ping_df.head())
print("********* Head of the test_df**************")
print(test_df.head())
```

```
shape of the driver_df: (2500, 4)
shape of the ping_df: (50528701, 2)
shape of the test_df: (17500, 3)
********* Head of the driver_df**************
   driver_id gender  age  number_of_kids
0     979863   MALE   26               2
1     780123   MALE   60               2
2     614848   MALE   45               4
3     775046   MALE   62               3
4     991601   MALE   23               0
********* Head of the ping_df**************
   driver_id  ping_timestamp
0     899313      1496278800
1     373017      1496278800
2     798984      1496278800
3     245966      1496278800
4     689783      1496278800
********* Head of the test_df**************
   driver_id       date  online_hours
0     979863  2017-6-28             7
1     979863  2017-6-27             9
2     979863  2017-6-26             9
3     979863  2017-6-25            10
4     979863  2017-6-24             9
```

```
#check for the null values in driver dataset
driver_df.isnull().sum()
```

```
driver_id         0
gender            0
age               0
number_of_kids    0
dtype: int64
```

```
#check for the null values in ping dataset
ping_df.isnull().sum()
```

```
driver_id         0
ping_timestamp    0
dtype: int64
```

```
driver_df.describe()
```

	driver_id	age	number_of_kids
count	2500.000000	2500.000000	2500.000000
mean	562397.047200	35.922400	1.395200
std	256410.208166	14.171207	1.505697
min	111556.000000	18.000000	0.000000
25%	343199.000000	25.000000	0.000000
50%	563854.500000	31.000000	1.000000
75%	787978.750000	45.000000	3.000000
max	998740.000000	75.000000	4.000000

```
ping_df = ping_df.sort_values(by=["driver_id","ping_timestamp"]).reset_index(drop = True)
ping_df.head()
```

	driver_id	ping_timestamp
0	111556	1496279340
1	111556	1496279355
2	111556	1496279370
3	111556	1496279400
4	111556	1496279430

```
from datetime import datetime
#create a copy of ping_df
temp_ping_df = ping_df.copy()

#Preprocessing_data
temp_ping_df.drop_duplicates(inplace=True) #remove duplicate data

#convert unixtimestamp into datetime
temp_ping_df["datetime"] = temp_ping_df["ping_timestamp"].apply(lambda x: datetime.fromtim

temp_ping_df.head()
```

	driver_id	ping_timestamp	datetime
0	111556	1496279340	2017-06-01 01:09:00
1	111556	1496279355	2017-06-01 01:09:15
2	111556	1496279370	2017-06-01 01:09:30

```
temp_ping_df["date"] = temp_ping_df["datetime"].dt.date
temp_ping_df.head()
```

	driver_id	ping_timestamp	datetime	date
0	111556	1496279340	2017-06-01 01:09:00	2017-06-01
1	111556	1496279355	2017-06-01 01:09:15	2017-06-01
2	111556	1496279370	2017-06-01 01:09:30	2017-06-01
3	111556	1496279400	2017-06-01 01:10:00	2017-06-01
4	111556	1496279430	2017-06-01 01:10:30	2017-06-01

```
'''we need to convert the timestamp into online hours so take one step time difference for
and assumed that if the difference is less than 2 minutes then driver online hours are sam
it to 2 minutes'''
temp_ping_df['online_hours'] = (temp_ping_df.groupby(by=['driver_id','date'])['ping_timest
temp_ping_df.head()
```

	driver_id	ping_timestamp	datetime	date	online_hours
0	111556	1496279340	2017-06-01 01:09:00	2017-06-01	NaN
1	111556	1496279355	2017-06-01 01:09:15	2017-06-01	0.004167
2	111556	1496279370	2017-06-01 01:09:30	2017-06-01	0.004167
3	111556	1496279400	2017-06-01 01:10:00	2017-06-01	0.008333
4	111556	1496279430	2017-06-01 01:10:30	2017-06-01	0.008333

```
#here we don't have any limit
temp_ping_df['online_hours']  =  temp_ping_df['online_hours'].apply(lambda x: x if x< (2/6
temp_ping_df
```

	driver_id	ping_timestamp	datetime	date	online_hours
0	111556	1496279340	2017-06-01 01:09:00	2017-06-01	0.033333
1	111556	1496279355	2017-06-01 01:09:15	2017-06-01	0.004167
2	111556	1496279370	2017-06-01 01:09:30	2017-06-01	0.004167
3	111556	1496279400	2017-06-01 01:10:00	2017-06-01	0.008333
4	111556	1496279430	2017-06-01 01:10:30	2017-06-01	0.008333

```
temp_ping_df.fillna(0,inplace = True)

#### creating our training data
train_df= (temp_ping_df.groupby(by = ['driver_id','date'])['online_hours'].sum()).reset_in
train_df.head(10)
```

	driver_id	date	online_hours
0	111556	2017-06-01	2.250000
1	111556	2017-06-02	2.533333
2	111556	2017-06-05	4.700000
3	111556	2017-06-06	3.150000
4	111556	2017-06-07	2.662500
5	111556	2017-06-08	3.212500
6	111556	2017-06-09	4.466667
7	111556	2017-06-12	3.766667
8	111556	2017-06-13	4.416667
9	111556	2017-06-14	2.016667

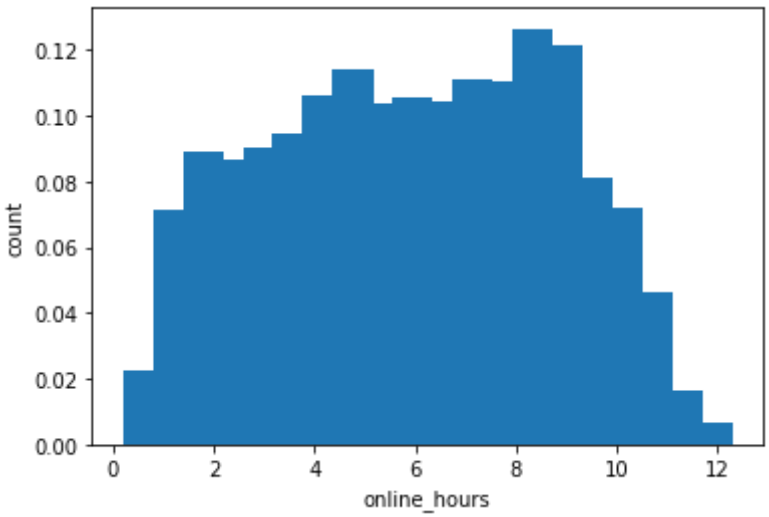
```
train_df['online_hours'] = round(train_df['online_hours'],1)
train_df.head(10)
```

	driver_id	date	online_hours
0	111556	2017-06-01	2.3

```
#Data Visualization
counts,bins = np.histogram(train_df["online_hours"],bins = 20,density =True)
print(counts)
```

```
[0.02234121 0.07166126 0.08912289 0.0869049  0.09029238 0.09456705
 0.10630224 0.114287   0.10384229 0.10533439 0.10452785 0.11098018
 0.11045593 0.12646579 0.12134425 0.08113813 0.07190322 0.04625519
 0.01621149 0.00673463]
```

```
plt.bar(bins[1:],counts)
plt.xlabel("online_hours")
plt.ylabel("count")
plt.show()
```



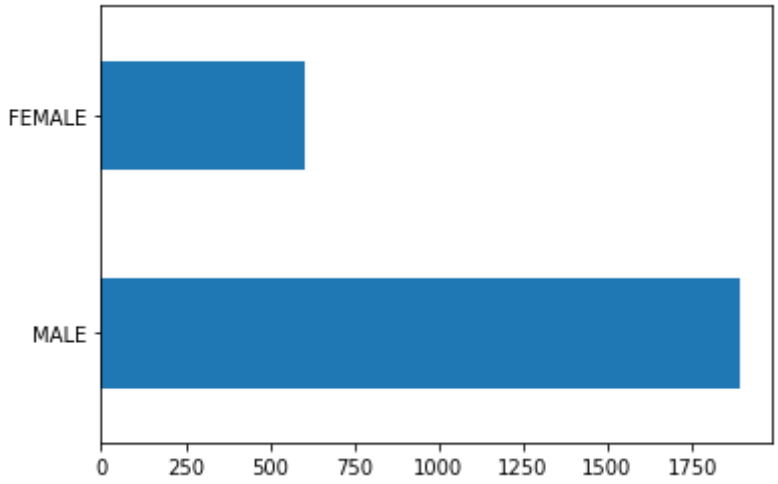
this is the histogram of online hours. we can see that the maximum density lies between 4 hrs to 8 hrs.

```
sns.violinplot("online_hours",data = train_df)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```

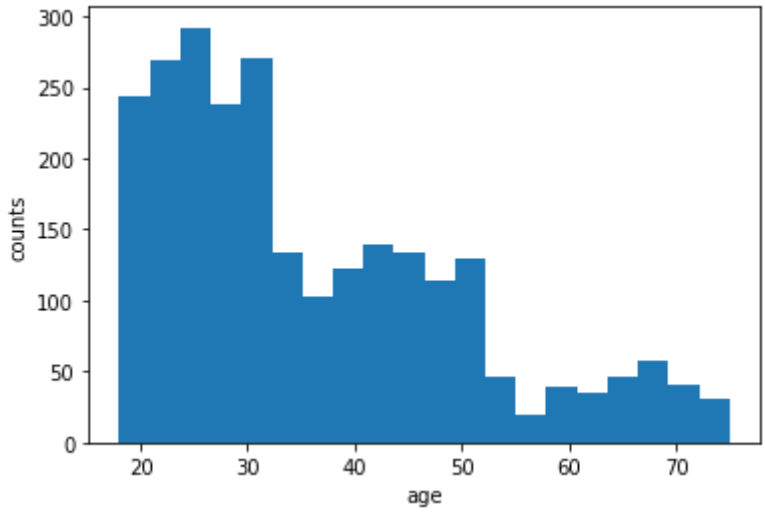
so most of the drivers are available from 4 hrs to 9 hrs.

```
driver_df["gender"].value_counts().plot(kind="barh")
plt.show()
```



above plot shows the No of male and female drivers. here we can see than no of male drivers are approx 3 times more than the no. of female drivers.

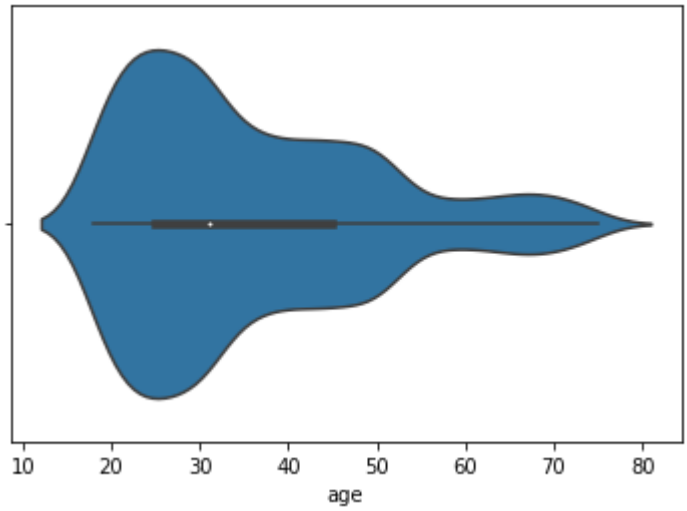
```
plt.hist(driver_df["age"],bins = 20)
plt.xlabel("age")
plt.ylabel("counts")
plt.show()
```



this is the histogram of the age. we can clearly see that the maximum drivers are having age in the range of 20 to 35.

```
sns.violinplot("age",data = driver_df)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```



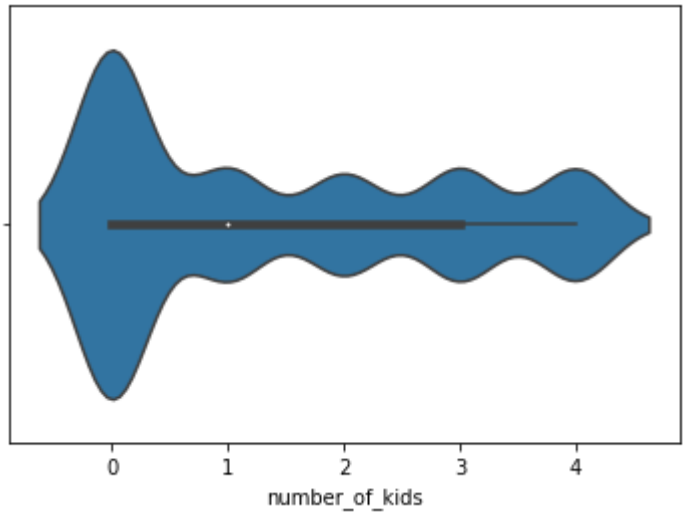
approx 50% drivers are having age less than 35.

```
plt.figure(figsize = (15,15))
sns.pairplot(driver_df)
plt.show()
```

```
<Figure size 1080x1080 with 0 Axes>
```

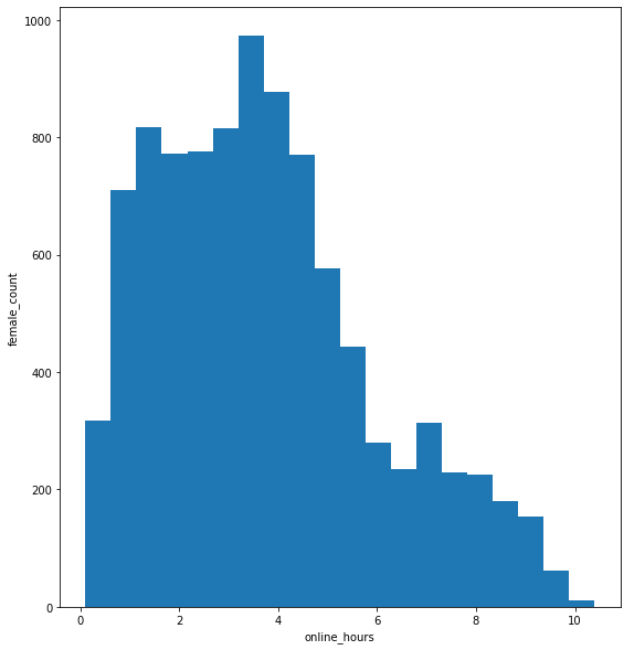
```
sns.violinplot("number_of_kids",data = driver_df)
plt.show()
```

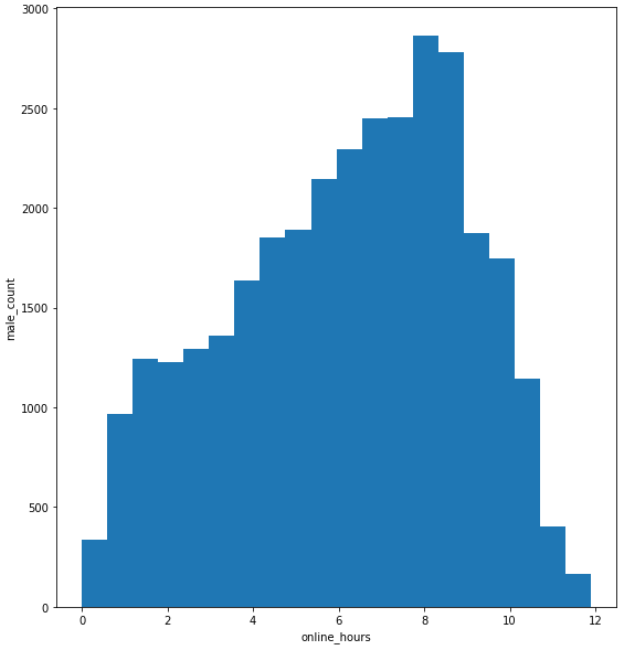
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```



Most of the drivers don't have any kids.

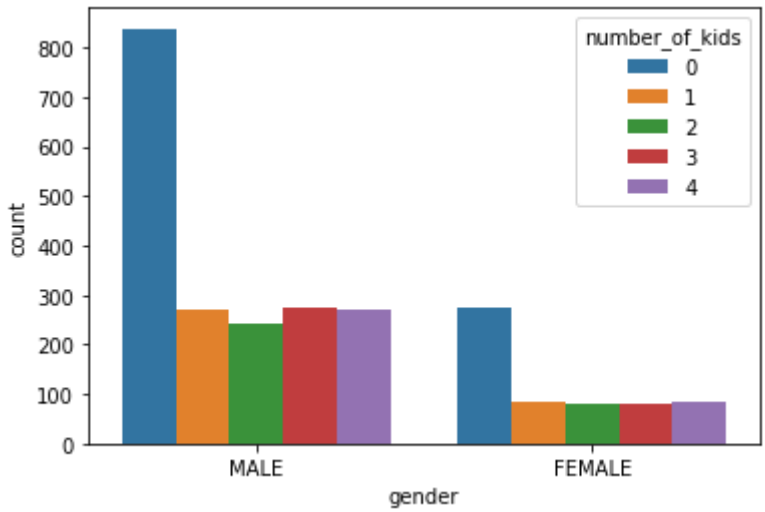
```
#lets check for the online availibility of male and female drivers
male_driver_ids = driver_df[driver_df["gender"]=="MALE"].driver_id
female_driver_ids = driver_df[driver_df["gender"]=="FEMALE"].driver_id
female_online_hours = train_df[train_df["driver_id"].isin(female_driver_ids)]["online_hour
male_online_hours = train_df[train_df["driver_id"].isin(male_driver_ids)]["online_hours"]


plt.figure(figsize=(20,10))
plt.subplot(121)
plt.hist(female_online_hours,bins = 20)
plt.xlabel("online_hours")
plt.ylabel("female_count")
plt.subplot(122)
plt.hist(male_online_hours,bins = 20)
plt.xlabel("online_hours")
plt.ylabel("male_count")
plt.show()
```



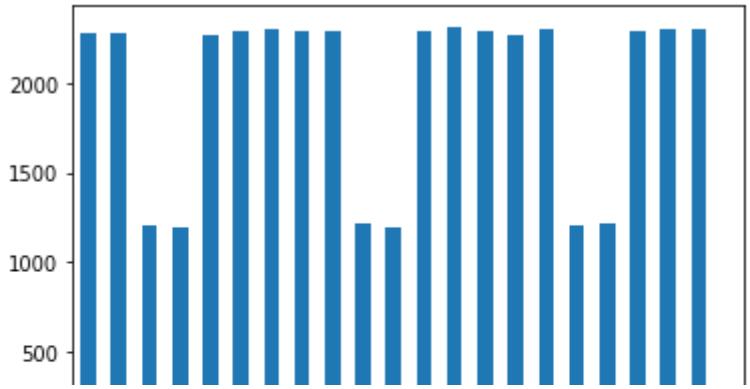


most of the female driver's online hours are in range of 1 to 5 and in case of male driver's that is in the range of 6 to 8. so from the data we can say the availibilty of male driver is more as compared to female.

```
sns.countplot(x = "gender",hue="number_of_kids",data=driver_df)
plt.show()
```




```
train_df.groupby("date").driver_id.count().plot(kind="bar")
plt.show()
```



here we can clearly see that on saturdays and sundays driver availibilty goes down by 50% approx.

```
'''we have data of 22nd june in the train data but we need to predict that in testing so w
the data is not balanced so we need to add 0 online hours for some dates.'''

train_df["date"] = train_df["date"].astype(str)
train_df = train_df[train_df["date"]<'2017-06-22'] ## data correction

train_date_df = pd.DataFrame({'date':pd.date_range(start='2017-06-01',end = '2017-06-21')}
train_date_df.head()
```

	date
0	2017-06-01
1	2017-06-02
2	2017-06-03
3	2017-06-04
4	2017-06-05

```
train_date_df["date"] = (train_date_df["date"]).astype(str)

ids_group = train_df.groupby("driver_id")["date"].unique()
ids_group.head()
```

```
driver_id
111556    [2017-06-01, 2017-06-02, 2017-06-05, 2017-06-0...
111575    [2017-06-01, 2017-06-05, 2017-06-06, 2017-06-0...
111779    [2017-06-01, 2017-06-02, 2017-06-06, 2017-06-0...
111839    [2017-06-01, 2017-06-02, 2017-06-03, 2017-06-0...
112486    [2017-06-01, 2017-06-02, 2017-06-05, 2017-06-0...
Name: date, dtype: object
```

```
driver_id_list = []
date_list = []
online_hours = []
for ids,dates in zip(ids_group.keys(),ids_group.values):
    date_ = dates.tolist()
```

```
    for date in train_date_df["date"].tolist():
        if date_ == train_date_df["date"].tolist():
          break
        else:
          if date not in date_:
            driver_id_list.append(ids)
            date_list.append(date)
            online_hours.append(0)

zero_df = pd.DataFrame({'driver_id':driver_id_list,'date':date_list,'online_hours':online_

zero_df.head()
```

	driver_id	date	online_hours
0	111556	2017-06-03	0
1	111556	2017-06-04	0
2	111556	2017-06-10	0
3	111556	2017-06-11	0
4	111556	2017-06-16	0

```
#train_df_backup = train_df.copy()

train_df = pd.concat([train_df,zero_df],sort =False)

train_df = train_df.sort_values(by=["driver_id","date"])
```

```
#check for any duplicate data in train_df

print("any duplicate values in the training data")
print(train_df.duplicated().any(),"\n")

print("any null values in the train_df")
print(train_df.isnull().any())
```

```
any duplicate values in the training data
False

any null values in the train_df
driver_id       False
date            False
online_hours    False
dtype: bool
```

```
train_df.head()
```

	driver_id	date	online_hours
0	111556	2017-06-01	2.3
1	111556	2017-06-02	2.5
0	111556	2017-06-03	0.0
1	111556	2017-06-04	0.0
2	111556	2017-06-05	4.7

```
train_df.shape
```

```
(52080, 3)
```

```
#concatenate the train and test data
test_df = test_df.sort_values(by = ["driver_id","date"])
test_df["date"] = pd.to_datetime(test_df["date"]).dt.date
test_df_backup = test_df.copy()
#test_dropped_df = test_df.drop("online_hours",axis =1)
df = pd.concat([train_df, test_df], sort=False)
print('\n' + 'Shape of The Concatinated DateFrame: {}'.format(df.shape) +'\n')

print('\n' +'*'*10+ ' Head of The Concatinated DataFrame ' +'*'*10+'\n')
print(df.head())

print('\n' +'*'*10+ ' Tail of The Concatinated DataFrame ' +'*'*10+'\n')
print(df.tail())

print('\n' +'*'*5+ ' After filling NaN values with Zeros in Concatinated DataFrame ' +'*'*
df.fillna(0,inplace = True)
print(df.tail())
print('\n' +'*'*50+ '\n')
```

```
Shape of The Concatinated DateFrame: (69580, 3)


********** Head of The Concatinated DataFrame **********

   driver_id        date  online_hours
0     111556  2017-06-01           2.3
1     111556  2017-06-02           2.5
0     111556  2017-06-03           0.0
1     111556  2017-06-04           0.0
2     111556  2017-06-05           4.7

********** Tail of The Concatinated DataFrame **********

       driver_id        date  online_hours
12387     998740  2017-06-24           0.0
12386     998740  2017-06-25           0.0
12385     998740  2017-06-26           0.0
12384     998740  2017-06-27           0.0
12383     998740  2017-06-28           0.0
```

```
***** After filling NaN values with Zeros in Concatinated DataFrame *****

       driver_id        date  online_hours
12387     998740  2017-06-24           0.0
12386     998740  2017-06-25           0.0
12385     998740  2017-06-26           0.0
12384     998740  2017-06-27           0.0
12383     998740  2017-06-28           0.0

***********************************************
```

```
temp_df = pd.merge(left = driver_df, right = df, on = 'driver_id', how = 'right')

temp_df.dropna(inplace = True) #drop null values
temp_df['gender'] = temp_df['gender'].replace({'MALE':1, 'FEMALE':0}) #change categorical

temp_df['date'] = pd.to_datetime(temp_df['date'])

## data and time related basic features
temp_df['day_name'] = temp_df['date'].dt.day_name()
temp_df['day'] = temp_df['date'].dt.day
temp_df['month'] = temp_df['date'].dt.month
temp_df['month_name'] = temp_df['date'].dt.month_name()
temp_df['year'] = temp_df['date'].dt.year
```

```
week_names = {'Sunday':0,'Monday':'1','Tuesday':2,'Wednesday':3, 'Thursday':4,'Friday':5,'
month_names = {'January':0, 'February':1,'March':2,'April':3,'May':4,'June':5,'July':6,
               'August':7, 'September':8,'October':9,'November':10,'December':11}

temp_df['day_name'] = temp_df['day_name'].map(week_names)
temp_df['month_name'] = temp_df['month_name'].map(month_names)
```

```
temp_df.head()
```

	driver_id	gender	age	number_of_kids	date	online_hours	day_name	day	month
0	111556	0	49	4	2017-06-01	2.3	4	1	6
1	111556	0	49	4	2017-06-02	2.5	5	2	6
2	111556	0	49	4	2017-06-03	0.0	6	3	6
3	111556	0	49	4	2017-	0.0	0	4	6

```
temp_df.tail()
```

	driver_id	gender	age	number_of_kids	date	online_hours	day_name	day	mo
69680	998740	1	27	0	2017-06-24	0.0	6	24	
69681	998740	1	27	0	2017-06-25	0.0	0	25	
69682	998740	1	27	0	2017-	0.0	1	26	

```
from sklearn.model_selection import (TimeSeriesSplit,
                                     GridSearchCV,
                                     RandomizedSearchCV,
                                     train_test_split,
                                     KFold,
                                     StratifiedKFold,
                                    cross_val_score)

from sklearn.preprocessing import (LabelEncoder,
                                   StandardScaler,
                                   MinMaxScaler,
                                   OrdinalEncoder)

from sklearn.feature_selection import SelectFromModel


# metrics
from sklearn.metrics import (mean_squared_error,
                             r2_score,
                             mean_absolute_error)
from sklearn.metrics import make_scorer


# modeling algos
from sklearn.linear_model import (LogisticRegression,
                                  Lasso,
                                  ridge_regression,
                                  LinearRegression)
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import (AdaBoostRegressor,
                              RandomForestRegressor,
                              VotingRegressor,
                              GradientBoostingRegressor)
from xgboost import XGBRegressor
from lightgbm import (LGBMRegressor,
                      early_stopping)

from sklearn.base import clone ## sklearn base models for stacked ensemble model



### solving model like a typical regression problem without any considerations and feature

# train and test data spliting
train = temp_df[temp_df['day'] < 22]
val = temp_df[temp_df['day'] >= 22]
```

```
print("train_shape:",train.shape)
print("val_shape:",val.shape)

X_train,X_test = train.drop(columns = ['online_hours','date']),val.drop(columns = ['online
y_train,y_test = train['online_hours'].values, val['online_hours'].values

scaler = MinMaxScaler() #Normalize the data.

X_train = scaler.fit_transform(X_train)
X_test  = scaler.transform(X_test)
## appending all the regressors

regressors = []

regressors.append(LinearRegression())
regressors.append(Lasso(alpha = 0.0005,max_iter = 1000))
regressors.append(RandomForestRegressor(n_estimators = 1000,max_depth=15))
regressors.append(AdaBoostRegressor())
regressors.append(GradientBoostingRegressor())
regressors.append(XGBRegressor(n_estimators = 1000,importance_type = 'gain'))
regressors.append(LGBMRegressor(n_estimators = 1000, objective ='regression',
                                importance_type = 'gain'))


import time

print('Working on base models and fitting begains.....')



baseline_models= clone(regressors)
algo = ['LinearRegression','Lasso', 'RandomForestRegressor' ,'AdaBoostRegressor',
                                          'GradientBoostingRegressor', 'XGBRegressor','LGB
rmse_list = []
for idx, reg in enumerate(baseline_models):
    t = time.time()
    print('Fitting of {} Model'.format(algo[idx]))
    print('Parameters of the model are: {}'. format(reg.get_params()))

    model = reg
    model.fit(X_train,y_train)
    preds = model.predict(X_test)
    rmse = mean_squared_error(y_test,preds,squared = False)

    print('time elapsed is : {} sec'.format(round((time.time() - t),2)))
    print('\n\n\n*****************************\n\n\n')
    rmse_list.append(rmse)

baseline = pd.DataFrame({'rmse':rmse_list,
                      "Algorithm":['LinearRegression','Lasso', 'RandomForestRegressor' ,'A
                                          'GradientBoostingRegressor', 'XGBRegressor','LGB
```

```
 train_shape: (52143, 11)
 val_shape: (17542, 11)
```

```
Working on base models and fitting begains.....
Fitting of LinearRegression Model
Parameters of the model are: {'copy_X': True, 'fit_intercept': True, 'n_jobs': Non
time elapsed is : 0.02 sec



****************************



Fitting of Lasso Model
Parameters of the model are: {'alpha': 0.0005, 'copy_X': True, 'fit_intercept': Tr
time elapsed is : 0.01 sec



****************************



Fitting of RandomForestRegressor Model
Parameters of the model are: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'm
time elapsed is : 96.3 sec



****************************



Fitting of AdaBoostRegressor Model
Parameters of the model are: {'base_estimator': None, 'learning_rate': 1.0, 'loss'
time elapsed is : 0.66 sec



****************************



Fitting of GradientBoostingRegressor Model
Parameters of the model are: {'alpha': 0.9, 'ccp_alpha': 0.0, 'criterion': 'friedm
time elapsed is : 3.73 sec



****************************



Fitting of XGBRegressor Model
Parameters of the model are: {'base_score': 0.5, 'booster': 'gbtree', 'colsample_by
[06:25:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is
time elapsed is : 19.49 sec
```

baseline

	rmse	Algorithm
0	3.227856	LinearRegression
1	3.227618	Lasso
2	2.472557	RandomForestRegressor
3	3.165607	AdaBoostRegressor
4	3.081979	GradientBoostingRegressor
5	2.758674	XGBRegressor
6	2.390156	LGBMRegressor

here we can see that the LGBMRegressor is giving least RMSE score.

```
error = pd.DataFrame({
    "date":val.date,
    "id":val.driver_id,
    "actual_online_hours":y_test.tolist(),
    "pred":preds
}).reset_index(drop = True)

error["error"] = np.abs(error.actual_online_hours-error.pred)

error.sort_values("error").head(20)
```

	date	id	actual_online_hours	pred	error
4298	2017-06-22	340766	3.0	2.999956	0.000044
6227	2017-06-26	431542	6.0	5.999745	0.000255
2187	2017-06-25	233165	0.0	-0.000634	0.000634
1746	2017-06-25	211582	6.0	5.999270	0.000730

```
# Feature Engineering
def datetime_features(data):
    data['date'] = pd.to_datetime(data['date'])
    data['month'] = data.date.dt.month
    data['day_of_month'] = data.date.dt.day
    data['day_of_year'] = data.date.dt.dayofyear
    data['week_of_year'] = data.date.dt.weekofyear
    data['day_of_week'] = data.date.dt.dayofweek + 1
    data['year'] = data.date.dt.year
    data["is_wknd"] = data.date.dt.weekday // 4
    data["quarter"] = data.date.dt.quarter
    data['is_month_start'] = data.date.dt.is_month_start.astype(int)
    data['is_month_end'] = data.date.dt.is_month_end.astype(int)
    data['is_quarter_start'] = data.date.dt.is_quarter_start.astype(int)
    data['is_quarter_end'] = data.date.dt.is_quarter_end.astype(int)
    data['is_year_start'] = data.date.dt.is_year_start.astype(int)
    data['is_year_end'] = data.date.dt.is_year_end.astype(int)

    week_names = {'Sunday':0,'Monday':1,'Tuesday':2,'Wednesday':3, 'Thursday':4,'Friday':5
    month_names = {'January':0, 'February':1,'March':2,'April':3,'May':4,'June':5,'July':6
                'August':7, 'September':8,'October':9,'November':10,'December':11}

    return data


def rolling_window_mean(data):
    # 7 day rolling window mean
    for i in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,61,17,18,19,20,21]:
        data["onlinehours_roll_mean_"+str(i)] = data.groupby(["driver_id"])['online_hours'
    #data.fillna(0,inplace = True)
    return data

def lag_features(data,lags = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,61,17,18,19,20,21]):
    data.sort_values(by=['driver_id','date'], axis=0, inplace=True)
    data = data.copy()
    for lag in lags:
        data['last_day_onlinehours'+str(lag)] = data.groupby('driver_id')['online_hours'].
        data['last_day_hours_diff' +str(lag)] = data.groupby('driver_id')['last_day_online
        #dataframe.fillna(0,inplace = True)
    return data
def preprocessing_traindata(df):
    data = df.copy()
    data = datetime_features(data)
    data = rolling_window_mean(data)
    data = lag_features(data)
    return data
```

```
df_copy  = preprocessing_traindata(df)
df_copy_backup = df.copy()

df_copy = pd.merge(left = df_copy,right = driver_df,on = "driver_id",how = "left")

df_copy["gender"] = df_copy["gender"].replace({"MALE":1,"FEMALE":0})
train_final = df_copy[df_copy['day_of_month'] < 22]
test_final = df_copy[df_copy['day_of_month'] >= 22]
print('\n'+'*'*15 + 'Shapes of Final Data ' + '*'*15+'\n' )

print('Shape of the Train Data: {}'.format(train_final.shape))
print('Shape of the Test Data: {}'.format(test_final.shape))


print('\n'+'*'*50 + '\n')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: FutureWarning: Series


***************Shapes of Final Data ***************

Shape of the Train Data: (52143, 83)
Shape of the Test Data: (17542, 83)

**************************************************
```

```
errors = []
for day in range(15,21):

    train = train_final[train_final['day_of_month'] < day]
    valid = train_final[train_final['day_of_month'] == day]

    xtrain,xtest = train.drop(columns = ['online_hours','date']),valid.drop(columns = ['on
    ytrain,ytest = train['online_hours'].values, valid['online_hours'].values

    model = LGBMRegressor(n_estimators = 1000)
    model.fit(xtrain,ytrain)
    preds = model.predict(xtest)

    score = mean_squared_error(valid['online_hours'].values,preds,squared = False)
    print('Trained upto Day %d and RMSE for %d day Prediction is %.5f' % (day, day+1,score
    errors.append(score)

print('\n' +'*'*20  +'Final Mean RMSE Score'+'*'*20 + '\n')
print('                  Mean RMSE Score:    %.5f' % np.mean(errors))
print('\n' +'*'*61 + '\n')
```

```
Trained upto Day 15 and RMSE for 16 day Prediction is 2.20878
Trained upto Day 16 and RMSE for 17 day Prediction is 2.06280
Trained upto Day 17 and RMSE for 18 day Prediction is 1.91828
Trained upto Day 18 and RMSE for 19 day Prediction is 1.69884
Trained upto Day 19 and RMSE for 20 day Prediction is 2.09313
Trained upto Day 20 and RMSE for 21 day Prediction is 2.02779
```

```
      ********************Final Mean RMSE Score********************

                        Mean RMSE Score:    2.00160

      ************************************************************
```

```
train = df_copy[df_copy['day_of_month'] < 22]
test = df_copy[df_copy['day_of_month'] >= 22]
```

```
xtrain,xtest = train.drop(columns = ['date','online_hours']),test.drop(columns = ['date','
ytrain,ytest = train['online_hours'].values, test['online_hours'].values

xtrain = scaler.fit_transform(xtrain)
xtest = scaler.transform(xtest)

model = LGBMRegressor(n_estimators = 1000)
model.fit(xtrain,ytrain)
preds = model.predict(xtest)

score = mean_squared_error(test.online_hours.tolist(),preds,squared=False)

print('\n' +'*'*20  +'Baseline Model RMSE Score'+'*'*20 + '\n')
print('                  RMSE Score:    %.5f' % score)
print('\n' +'*'*61 + '\n')
```

```
      ********************Baseline Model RMSE Score********************

                       RMSE Score:    1.90683

      ************************************************************
```

```
error = pd.DataFrame({
    "date":test.date,
    "id":test.driver_id,
    "actual_online_hours":test.online_hours.tolist(),
    "pred":preds
}).reset_index(drop = True)

error["error"] = np.abs(error.actual_online_hours-error.pred)

error.sort_values("error").head(20)
```

	date	id	actual_online_hours	pred	error
415	2017-06-24	133172	0.0	-0.000030	0.000030
73	2017-06-25	114890	0.0	-0.000275	0.000275
8860	2017-06-27	566447	6.0	5.999657	0.000343
2726	2017-06-25	258530	0.0	-0.000359	0.000359
14465	2017-06-25	852338	0.0	-0.000365	0.000365
6338	2017-06-25	437893	8.0	8.000454	0.000454
9737	2017-06-22	614442	5.0	4.998987	0.001013
7987	2017-06-22	525604	3.0	3.001197	0.001197
14654	2017-06-25	863345	7.0	7.001330	0.001330
5216	2017-06-23	378731	6.0	5.998623	0.001377
15045	2017-06-24	882676	0.0	-0.001454	0.001454
11965	2017-06-24	730697	0.0	0.001459	0.001459
7244	2017-06-28	484883	2.0	2.001622	0.001622
706	2017-06-28	151806	7.0	7.001800	0.001800
7412	2017-06-28	493432	6.0	5.998196	0.001804

```
!pip install eli5
```

```
Collecting eli5
  Downloading https://files.pythonhosted.org/packages/d1/54/04cab6e1c0ae535bec93f795c
     |████████████████████████████████| 112kB 5.0MB/s
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from el
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-pack
Installing collected packages: eli5
Successfully installed eli5-0.11.0
```

```
#import shap
import eli5
from eli5.sklearn import PermutationImportance

train = train_final[train_final['day_of_month'] < 16]
valid = train_final[train_final['day_of_month'] >=16]

xtrain,xvalid = train.drop(columns = ['online_hours','date']),valid.drop(columns = ['onlin
ytrain,yvalid = train['online_hours'].values, valid['online_hours'].values
```

```
model = LGBMRegressor().fit(xtrain,ytrain)

perm = PermutationImportance(model,scoring='neg_root_mean_squared_error' ).fit(xvalid,yval
eli5_feature_importance = (pd.DataFrame({'Features':xtrain.columns.tolist(),'Importance':p
                           .sort_values(by = 'Importance'))

print('Feature impact weights and importance for the given model:')
eli5.show_weights(perm, feature_names = xtrain.columns.tolist())
```

Feature impact weights and importance for the given model:

Weight	Feature
1.2313 ± 0.0089	last_day_onlinehours7
0.1655 ± 0.0077	day_of_week
0.1432 ± 0.0033	last_day_onlinehours14
0.0986 ± 0.0068	last_day_onlinehours1
0.0475 ± 0.0070	age
0.0267 ± 0.0024	last_day_onlinehours6
0.0235 ± 0.0047	last_day_onlinehours4
0.0177 ± 0.0025	last_day_onlinehours3
0.0144 ± 0.0022	gender
0.0127 ± 0.0039	number_of_kids
0.0114 ± 0.0028	last_day_onlinehours5
0.0090 ± 0.0017	last_day_hours_diff6
0.0084 ± 0.0016	last_day_onlinehours10
0.0079 ± 0.0013	last_day_onlinehours2
0.0065 ± 0.0024	last_day_onlinehours9
0.0035 ± 0.0010	last_day_hours_diff13
0.0022 ± 0.0008	onlinehours_roll_mean_3
0.0020 ± 0.0007	last_day_hours_diff4
0.0018 ± 0.0023	last_day_hours_diff7
0.0016 ± 0.0005	last_day_onlinehours13

… 61 more …

```
## feature selection based on permutation importance
feature_import = (eli5_feature_importance.sort_values(by = 'Importance', ascending = False
                  .reset_index(drop = True).Features.tolist()[0:40])


train = df_copy[df_copy['day_of_month'] < 22]
test = df_copy[df_copy['day_of_month'] >= 22]


xtrain,xtest = train.drop(columns = ['date','online_hours']),test.drop(columns = ['date','
xtrain = xtrain[feature_import]
xtest = xtest[feature_import]
ytrain,ytest = train['online_hours'].values, test['online_hours'].values

model = LGBMRegressor(n_estimators = 1000)
model.fit(xtrain,ytrain)
preds = model.predict(xtest)

score = mean_squared_error(test.online_hours.tolist(),preds,squared=False)


print('\n' +'*'*20  +'Baseline Model RMSE Score with feature selection'+'*'*20 + '\n')
print('                 RMSE Score:    %.5f' % score)
print('\n' +'*'*81 + '\n')
```

```
print('\n' + '='*81 + '\n')
```

```
**********Baseline Model RMSE Score with feature selection**********

                RMSE Score:    1.88214

********************************************************************************
```

```
error = pd.DataFrame({
    "date":test.date,
    "id":test.driver_id,
    "actual_online_hours":test.online_hours.tolist(),
    "pred":preds
}).reset_index(drop = True)

error["error"] = np.abs(error.actual_online_hours-error.pred)

error.sort_values("error").head(30)
```

	date	id	actual_online_hours	pred	error
8843	2017-06-24	566319	0.0	-0.000039	0.000039
4832	2017-06-24	361795	0.0	0.000062	0.000062
3335	2017-06-25	287717	0.0	0.000203	0.000203
16806	2017-06-28	963766	7.0	7.000339	0.000339
3810	2017-06-24	313810	0.0	-0.000434	0.000434
9656	2017-06-25	607094	0.0	0.000453	0.000453
8948	2017-06-24	570715	0.0	-0.000490	0.000490
2180	2017-06-25	232934	0.0	-0.000521	0.000521
6590	2017-06-25	451132	0.0	-0.000597	0.000597
10118	2017-06-25	630713	0.0	-0.000665	0.000665
12812	2017-06-24	773817	0.0	-0.000760	0.000760
4119	2017-06-25	327050	0.0	-0.000791	0.000791
7412	2017-06-28	493432	6.0	5.999171	0.000829
10862	2017-06-27	668780	3.0	3.000864	0.000864
8473	2017-06-25	547254	0.0	0.000891	0.000891
10964	2017-06-24	675439	0.0	-0.000921	0.000921
15226	2017-06-23	889001	2.0	2.000933	0.000933
11191	2017-06-27	684092	6.0	5.999052	0.000948
11643	2017-06-24	711123	0.0	-0.001012	0.001012
16032	2017-06-24	927446	0.0	0.001029	0.001029
16821	2017-06-22	964765	6.0	6.001237	0.001237
4462	2017-06-25	344949	9.0	8.998714	0.001286
12063	2017-06-24	733929	0.0	-0.001400	0.001400

✓ 0s completed at 3:14 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.