In [640]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import seaborn as sns
import datetime
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

In [14]:

```
#Specifying the file details
path='/Users/s0p00zp/.kaggle/DS TakeHome Assignment/data'
pings='pings.csv'
driver='drivers.csv'
test='test.csv'
```

In [1148]:

```
pings_df=pd.read_csv(path+'/'+pings)
drivers_df=pd.read_csv(path+'/'+driver)
test_df=pd.read_csv(path+'/'+test)
```

Exploring Drivers data

In [23]:

```
drivers_df.head()
```

Out[23]:

	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

In [96]:

```
#checking number of unique driver ID's
len(set(drivers_df.driver_id))
```

Out[96]:

```
2497
```

In [109]:

```
#checking for missing values in data
drivers_df.isna().any()
```

Out[109]:

```
driver_id          False
gender             False
age                False
number_of_kids     False
dtype: bool
```

In [95]:

```
drivers_df.shape
```

Out[95]:

```
(2500, 4)
```

In [56]:

```
drivers_df.dtypes
```

Out[56]:

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

In [49]:

```
drivers_df.describe()
```

Out[49]:

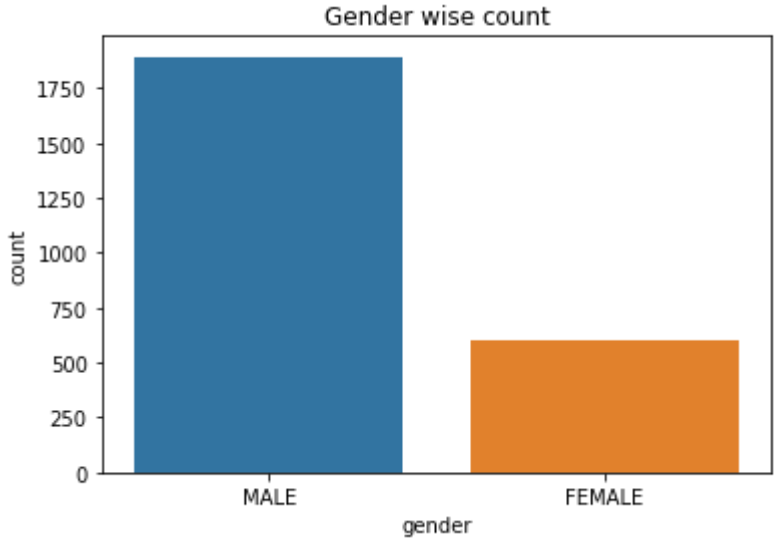
	driver_id	age	number_of_kids
count	2500.000000	2500.000000	2500.000000
mean	562397.047200	35.922400	1.395200
std	256410.208166	14.171207	1.505697
min	111556.000000	18.000000	0.000000
25%	343199.000000	25.000000	0.000000
50%	563854.500000	31.000000	1.000000
75%	787978.750000	45.000000	3.000000
max	998740.000000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

In [22]:

```
#exploring Drivers Data
sns.countplot(drivers_df.gender).set_title('Gender wise count')
```

Out[22]:

```
Text(0.5,1,'Gender wise count')
```

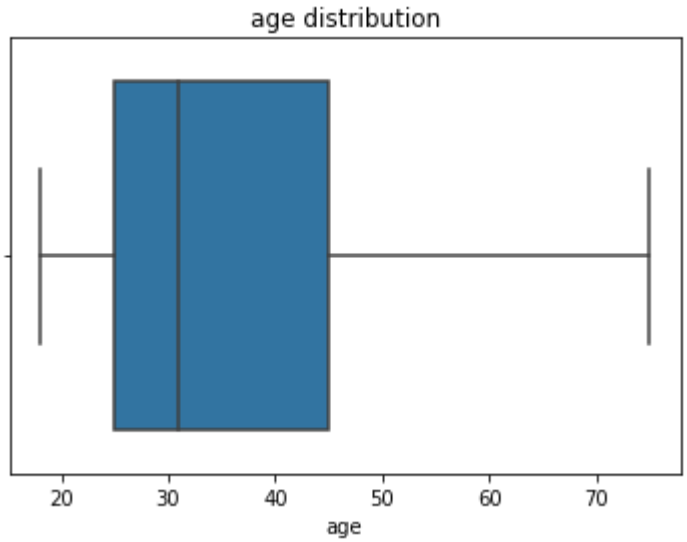


In [40]:

```
sns.boxplot(drivers_df.age).set_title('age distribution')
```

Out[40]:

```
Text(0.5,1,'age distribution')
```

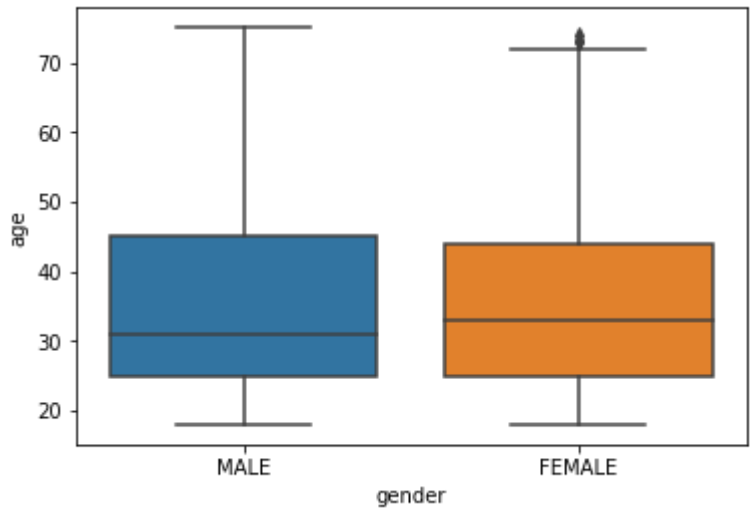


In [41]:

```
sns.boxplot(drivers_df.gender,drivers_df.age).set_title('gennder vs age')
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a36ab4160>
```



In [1145]:

```
print("75th percentile for female age:",drivers_df['age'][drivers_df['gender']==
'FEMALE'].quantile(0.75))
print("90th percentile for female age:",drivers_df['age'][drivers_df['gender']==
'FEMALE'].quantile(0.90))
print("99th percentile for female age:",drivers_df['age'][drivers_df['gender']==
'FEMALE'].quantile(0.99))
print("high numbers for female age:\n",drivers_df['age'][(drivers_df['gender']==
'FEMALE')&(drivers_df['age']>71)])
```

```
75th percentile for female age: 44.0
90th percentile for female age: 50.0
99th percentile for female age: 70.95000000000005
high numbers for female age:
 432     74
731     72
929     72
1637    73
1795    73
Name: age, dtype: int64
```

In [54]:

```
pd.cut(drivers_df.age,3,).value_counts()
```

Out[54]:

```
(17.943, 37.0]    1549
(37.0, 56.0]       693
(56.0, 75.0]       258
Name: age, dtype: int64
```

Higher the age of the drivers Lower is the number.

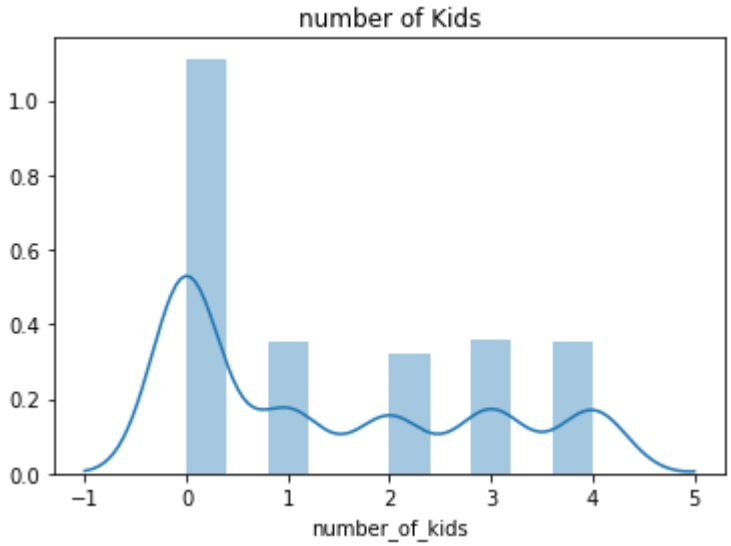
Younger age drivers are more

In [43]:

```
sns.distplot(drivers_df.number_of_kids).set_title('number of Kids')
```

Out[43]:

```
Text(0.5,1,'number of Kids')
```



In [46]:

```
pd.crosstab(drivers_df.gender,drivers_df.number_of_kids)
```

Out[46]:

number_of_kids	0	1	2	3	4
gender					
FEMALE	274	85	81	81	85
MALE	838	270	240	276	270

Exploring Pings data

In [92]:

```
pings_df.head()
```

Out[92]:

	driver_id	ping_timestamp
0	899313	1496278800
1	373017	1496278800
2	798984	1496278800
3	245966	1496278800
4	689783	1496278800

In [110]:

```
#datatypes in pings
pings_df.dtypes
```

Out[110]:

```
driver_id          int64
ping_timestamp     int64
dtype: object
```

In [97]:

```
pings_df.shape
```

Out[97]:

```
(50528701, 2)
```

In [99]:

```
print("number of Driver's id in ping data:",len(set(pings_df.driver_id)))
```

```
number of Driver's id in ping data: 2480
```

In [100]:

```
#checking differnce between drivers in driver data and ping data
print("Number of driver Ids not present in pings:",len(set(drivers_df.driver_id)
-set(pings_df.driver_id)))
```

```
Number of driver Ids not present in pings: 17
```

In [108]:

```
#checking for any missing values
pings_df.isna().any()
```

Out[108]:

```
driver_id          False
ping_timestamp     False
dtype: bool
```

In [118]:

```
#converting unix epochs time stamp to datetime format

pings_df['date']=[time.strftime('%Y-%m-%d %H:%M:%S',time.gmtime(int(x)+25200)) f
or x in pings_df.ping_timestamp]
```

In [134]:

```
pings_df['date']=pd.to_datetime(pings_df['date'])
```

In [170]:

```
#creating features from date column
#pings_df['dayofyear']=pings_df['date'].dt.dayofyear
#pings_df['dayofweek']=pings_df['date'].dt.dayofweek
#pings_df['minute']=pings_df['date'].dt.minute
pings_df['hour']=pings_df['date'].dt.hour
#pings_df['seconds']=pings_df['date'].dt.second
pings_df['date_month']=pings_df['date'].dt.date
```

In [1020]:

```
def calculate_online_hours(pings_df):

    #creating features from date column
    pings_df['hour']=pings_df['date'].dt.hour
    pings_df['date_month']=pings_df['date'].dt.date

    #selecting columns which are relevant
    pings_df_s=pings_df[['driver_id','date','date_month','hour']]

    #1st grouping with respect to driver_id ,date and hour of that date.
    pings_df_time=pings_df_s.groupby(['driver_id','date_month','hour'])['date'].
count()
    #Calculating number of pings in a specific hours of aday
    pings_df_time=pings_df_time.reset_index()
    #pings_df_time['date']=pings_df_time['date']-1

    #Multiplying number of pings with 15 since each ping is at an interval of 15
 sec
    #dividing the sum of all seconds by 3600
    pings_df_hrs=pings_df_time.groupby(['driver_id','date_month'])['date'].apply
(lambda x:(x.sum()*15)/3600)
    pings_df_hrs=pings_df_hrs.reset_index()

    #Rounding of the hours to calculate online_hours
    #pings_df_hrs['online_hours']=pings_df_hrs['date'].round()
    pings_df_hrs['online_hours']=np.floor(pings_df_hrs['date'])
    #dropping the date column from the data frame
    pings_df_hrs=pings_df_hrs.drop(['date'],axis=1)

    #creating features for date
    pings_df_hrs['dayofyear']=pings_df_hrs['date_month'].dt.dayofyear
    pings_df_hrs['dayofweek']=pings_df_hrs['date_month'].dt.dayofweek
    pings_df_hrs['dayofmonth']=pings_df_hrs['date_month'].dt.day

    return(pings_df_hrs)
```

In [265]:

```
#pings_df_group=pings_df[pings_df['driver_id']==899313]
#pings_df_s=pings_df_s[(pings_df_s.driver_id==998229)|(pings_df_s.driver_id==162
703)]
```

In [975]:

```
#1st grouping with respect to driver_id ,date and hour of that date.
#Calculating number of pings in a specific hours of aday
pings_df_time=pings_df_s.groupby(['driver_id','date_month','hour'])['date'].coun
t()
pings_df_time=pings_df_time.reset_index()
pings_df_time['date']=pings_df_time['date']-1
```

In [976]:

```
pings_df_time.head()
```

Out[976]:

	driver_id	date_month	hour	date
0	111556	2017-06-01	8	181
1	111556	2017-06-01	9	215
2	111556	2017-06-01	10	85
3	111556	2017-06-02	8	135
4	111556	2017-06-02	9	212

In [977]:

```
#Multiplying number of pings with 15 since each ping is at an interval of 15 sec
#dividing the sum of all seconds by 3600
pings_df_hrs=pings_df_time.groupby(['driver_id','date_month'])['date'].apply(lam
bda x:(x.sum()*15)/3600)
pings_df_hrs=pings_df_hrs.reset_index()
```

In [978]:

```
#Rounding of the hours to calculate online_hours
pings_df_hrs['online_hours']=np.floor(pings_df_hrs['date'])
```

In [979]:

```
min(pings_df_hrs.online_hours)
```

Out[979]:

```
0.0
```

In [980]:

```
#dropping the date column from the data frame
pings_df_hrs=pings_df_hrs.drop(['date'],axis=1)
```

In [981]:

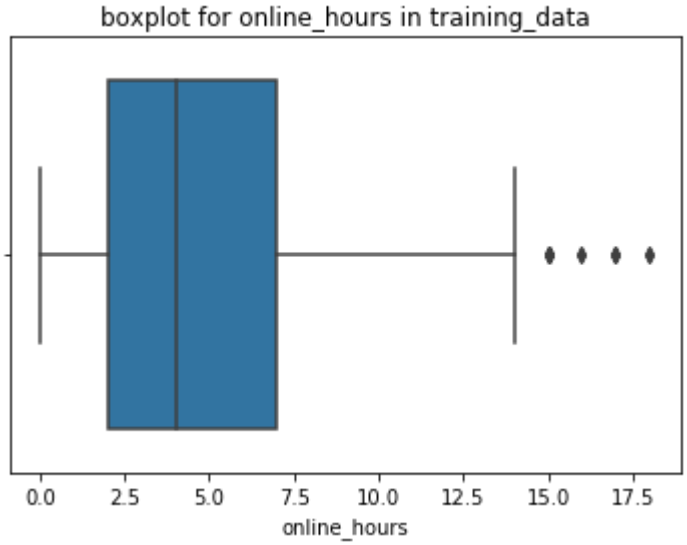
```
#creating features for date
pings_df_hrs['dayofyear']=pings_df_hrs['date_month'].dt.dayofyear
pings_df_hrs['dayofweek']=pings_df_hrs['date_month'].dt.dayofweek
pings_df_hrs['dayofmonth']=pings_df_hrs['date_month'].dt.day
```

In [1254]:

```
sns.boxplot(pings_df_hrs.online_hours).set_title('boxplot for online_hours in tr
aining_data')
```

Out[1254]:

```
Text(0.5,1,'boxplot for online_hours in training_data')
```

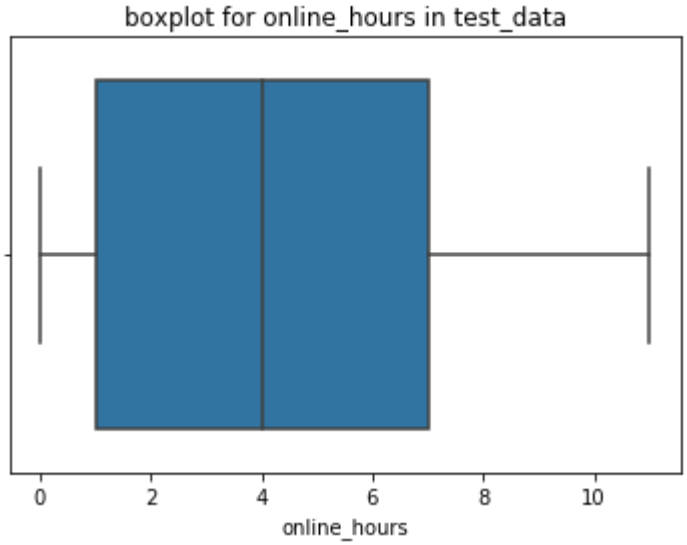


In [969]:

```
sns.boxplot(test_df.online_hours).set_title('boxplot for online_hours in test_da
ta')
```

Out[969]:

```
Text(0.5,1,'boxplot for online_hours in test_data')
```



1.distribution of online_hours in training data is different from test data.

2.There seems to be something different in training data.

3.Need further scanning of the training data

In [1149]:

```
#Checking for duplicate rows in training data
duplicate=pings_df.duplicated()
```

In [1150]:

```
#no of duplicate records
print("number of duplicate records:",np.sum(duplicate))
```

```
number of duplicate records: 39543
```

In [987]:

```
#removing duplicate records
pings_df_s_clean=pings_df_s[~duplicate]
#check duplicate and clean records
pings_df_s_clean.shape[0],pings_df_s.shape[0]
```

Out[987]:

```
(50489158, 50528701)
```

In [1021]:

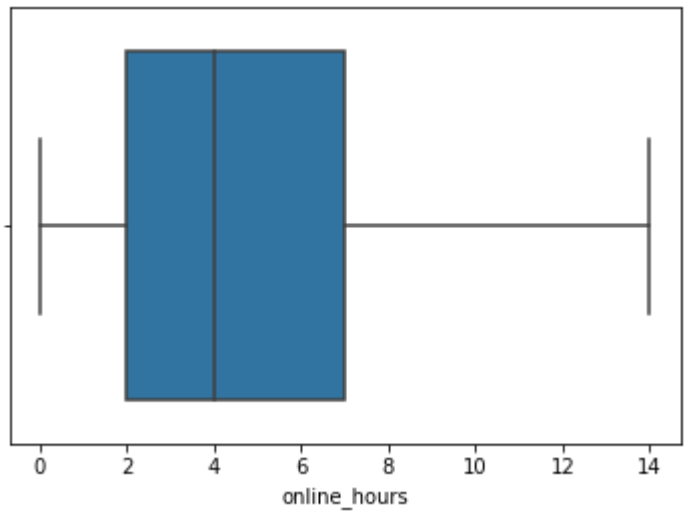
```
train_data=calculate_online_hours(pings_df_s_clean)
```

In [1022]:

```
sns.boxplot(train_data.online_hours)
```

Out[1022]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bc374b0f0>
```



In [1023]:

```
train_data[train_data.online_hours>13]
```

Out[1023]:

	driver_id	date_month	online_hours	dayofyear	dayofweek	dayofmonth
41623	988496	2017-06-12	14.0	163	0	12

Creating a Baseline model with only mean w.r.t driver_id

In [1249]:

```
#creating a baseline model by predicting online_hours as mean
def max_min(x):
    return(x.max()-x.min())
def percentile_1(x):
    return(np.percentile(x,0.25))
def percentile_2(x):
    return(np.percentile(x,0.75))
def percentile_3(x):
    return(np.percentile(x,0.50))
def percentile_4(x):
    return(np.percentile(x,0.95))
def percentile_5(x):
    return(np.percentile(x,0.10))
f={'online_hours':['mean','min','max',max_min,percentile_1,percentile_2,percenti
le_3,percentile_4,percentile_5]}

#f={'online_hours':['mean']}
#train_df_mean=train_data.groupby(['driver_id'])['online_hours'].mean()
train_df_mean=train_data.groupby(['driver_id']).agg(f)
train_df_mean=train_df_mean.reset_index()
```

In [1237]:

```
train_df_mean.columns=train_df_mean.columns.map('_'.join)
train_df_mean=train_df_mean.rename(columns={"driver_id_":"driver_id"})
#train_df_mean['online_hours_mean']=train_df_mean['online_hours_mean'].round()
```

In [1250]:

```
train_df_mean.columns=train_df_mean.columns.map('_'.join)
train_df_mean=train_df_mean.rename(columns={"driver_id_":"driver_id"})
train_df_mean.head()
```

Out[1250]:

	driver_id	online_hours_mean	online_hours_min	online_hours_max	online_hours_max_min	c
0	111556	2.000000	0.0	4.0	4.0	
1	111575	5.916667	3.0	8.0	5.0	
2	111779	2.666667	1.0	4.0	3.0	
3	111839	6.000000	1.0	8.0	7.0	
4	112486	2.466667	0.0	4.0	4.0	

In [1158]:

```
#train_df_mean=train_df_mean.drop(['index'],axis=1)
train_df_mean['online_hours_mean']=np.floor(train_df_mean['online_hours_mean'])
train_df_mean.head()
```

Out[1158]:

	driver_id	online_hours_mean	online_hours_min	online_hours_max	online_hours_max_min	c
0	111556	2.0	0.0	4.0	4.0	
1	111575	6.0	3.0	8.0	5.0	
2	111779	3.0	1.0	4.0	3.0	
3	111839	6.0	1.0	8.0	7.0	
4	112486	2.0	0.0	4.0	4.0	

In [1220]:

```
pred_base_y=pd.merge(test_df,train_df_mean,how='left',on='driver_id')
```

We have Null values in prediction since there are frew driver_id which are not present in Train

In [1221]:

```
pred_base_y.isna().any()
```

Out[1221]:

```
driver_id            False
date                 False
online_hours         False
ind                  False
online_hours_mean     True
dtype: bool
```

In [1222]:

```
#checking for Null values in pred_y
driver_not_in_training=np.unique(pred_base_y.driver_id[pred_base_y.online_hours_
mean.isna()])
Index_driver_not_in_training=[idx for idx,x in enumerate(test_df.driver_id) if x
 in list(driver_not_in_training)]
print(np.unique(pred_base_y.driver_id[pred_base_y.online_hours_mean.isna()]))
```

```
[230923 373792 425331 523243 585955 616243 675613 682678 743899 7566
84
 772057 808404 854976 934994 971478 993757 998740]
```

Imputing Null Values with mean and median

In [1223]:

```
#pred_base_y.online_hours_y=pred_base_y.online_hours_y.fillna(np.mean(train_dat
a.online_hours))
pred_base_y.online_hours_mean=pred_base_y.online_hours_mean.fillna(0)
```

In [1203]:

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [1243]:

```
#Mean Squared error for baseline model with only mean
error=mean_squared_error(pred_base_y.online_hours,pred_base_y.online_hours_mean)
print("MSE for baseline Model is :",error)
```

```
MSE for baseline Model is : 6.947257142857143
```

Adding driver details data

In [1001]:

```
#drivers details have additional 17 drivers data which is not present in ping da
ta
len(set(train_data.driver_id)),len(set(drivers_df.driver_id))
```

Out[1001]:

```
(2480, 2497)
```

In [1206]:

```
def Creating_train_test(train_data,test_df,drivers_df,label_encode=0,merge=0):
    train_data=train_data.rename(columns={'date_month':'date'})
    train_data['ind']='tr'
    test_df['ind']='te'
    all_data=pd.concat([train_data,test_df],axis=0)
    #merging training data with driver data
    all_data_M=pd.merge(all_data.drop_duplicates(),drivers_df.drop_duplicates(),
how='inner',on='driver_id')
    if merge==0:
        all_data_M=pd.merge(all_data_M,train_df_mean,how="left",on="driver_id")
        all_data_M=all_data_M.fillna(0)
        #all_data_M=all_data_M.drop(['online_hours_mean_y'],axis=1)
        '''         all_data_M.columns=['driver_id','date',
                    'online_hours','dayofyear',
                    'dayofweek','dayofmonth',
                    'ind','gender','age','number_of_kids','online_hours_mean',
                           'online_hours_min','online_hours_max','online_hours_m
ax_min',
                            'online_hours_percentile_1','online_hours_percentile
_2',
                           'online_hours_percentile_3','online_hours_percentile_
4',
                           'online_hours_percentile_5']
        '''
    if label_encode==0:
        x=pd.get_dummies(all_data_M,columns=['driver_id','gender','dayofweek'])
    else:
        x=pd.get_dummies(all_data_M,columns=['gender','dayofweek'])
        #x['driver_id']=LabelEncoder().fit_transform(all_data_M.driver_id)
    x=x.drop(['date'],axis=1)
    x_train=x[x.ind=='tr']
    x_test=x[x.ind=='te']
    x_train=x_train.drop(['ind'],axis=1)
    x_test=x_test.drop(['ind'],axis=1)
    return(x_train,x_test)
```

In [1251]:

```
x_train,x_test=Creating_train_test(train_data,test_df,drivers_df,label_encode=1,
merge=0)
```

In [1226]:

```
driver_not_in_training=set(test_df.driver_id).difference(set(train_data.driver_i
d))
```

In [1252]:

```
id=[id for id,x in enumerate(x_test.driver_id) if x in list(driver_not_in_traini
ng)]
```

In [1228]:

```
'''
independant=['driver_id','dayofyear',
             'dayofmonth','age',
             'number_of_kids','online_hours_mean',
             'online_hours_min','online_hours_max','online_hours_max_min',
             'online_hours_percentile_1','online_hours_percentile_2',
             'gender_FEMALE','gender_MALE',
             'dayofweek_0','dayofweek_1',
             'dayofweek_2','dayofweek_3',
             'dayofweek_4','dayofweek_5','dayofweek_6']
'''
independant=[x for x in x_train.columns if x not in ['online_hours']]
dependant=['online_hours']
```

In [1099]:

```
x_train_upper=x_train[x_train.online_hours<np.mean(x_train.online_hours)]
x_train_lower=x_train[x_train.online_hours>np.mean(x_train.online_hours)]
```

In [940]:

```
model_rf_lower=RandomForestRegressor(n_estimators=100)
model_rf_lower.fit(x_train_lower[independant],x_train_lower[dependant])
model_rf_upper=RandomForestRegressor(n_estimators=100)
model_rf_upper.fit(x_train_upper[independant],x_train_upper[dependant])
```

Out[940]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=Non
e,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start
=False)
```

In [941]:

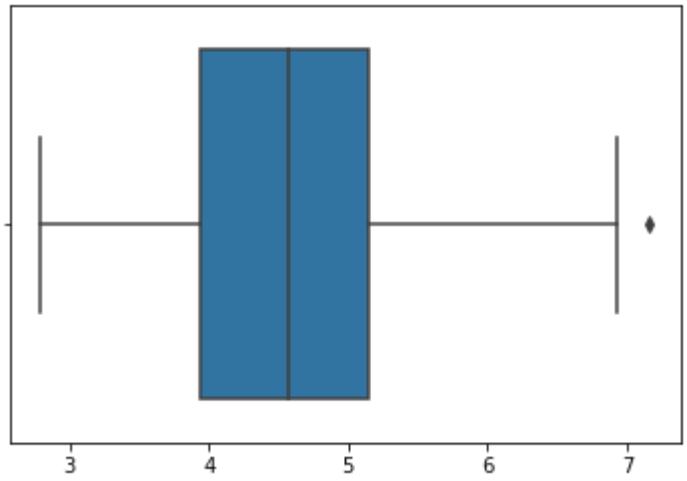
```
#training 2 model for onlinehours greater and less than mean
pred_lower=model_rf_lower.predict(x_test[independant])
pred_upper=model_rf_upper.predict(x_test[independant])
```

In [945]:

```
pfinal_pred=(pred_lower+pred_upper)/2
sns.boxplot(pfinal_pred)
```

Out[945]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bcf1c5c18>
```



In [875]:

```
model_rf1=RandomForestRegressor(n_estimators=700)
model_rf1.fit(x_train[independant],x_train[dependant])
model_rf2=RandomForestRegressor(n_estimators=500)
model_rf2.fit(x_train[independant],x_train[dependant])
model_rf3=RandomForestRegressor(n_estimators=1000)
model_rf3.fit(x_train[independant],x_train[dependant])
model_rf4=RandomForestRegressor(n_estimators=200)
model_rf4.fit(x_train[independant],x_train[dependant])
model_rf5=RandomForestRegressor(n_estimators=20)
model_rf5.fit(x_train[independant],x_train[dependant])
```

Out[875]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=Non
e,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start
=False)
```

In [1253]:

```
model_rf2=RandomForestRegressor(n_estimators=500)
model_rf2.fit(x_train[independant],x_train[dependant])
```

Out[1253]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=Non
e,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start
=False)
```

In [925]:

```
new_data=pd.DataFrame({'feature1':model_rf1.predict(x_train[independant]),
                       'feature2':model_rf2.predict(x_train[independant]),
                       'feature3':model_rf3.predict(x_train[independant]),
                       'feature4':model_rf4.predict(x_train[independant]),
                      'feature5':model_rf5.predict(x_train[independant])})
```

In [926]:

```
new_data.head()
```

Out[926]:

	feature1	feature2	feature3	feature4	feature5
0	2.144286	2.166	2.170	2.130	2.10
1	2.294286	2.278	2.345	2.400	2.30
2	3.267143	3.392	3.361	3.275	2.90
3	3.068571	2.998	3.060	3.030	3.15
4	2.230000	2.308	2.304	2.280	2.15

In [929]:

```
from sklearn.linear_model import LinearRegression
model_linear=LinearRegression()
model_linear.fit(new_data,x_train[dependant])
```

Out[929]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normaliz
e=False)
```

In [927]:

```
new_y=pd.DataFrame({'feature1':model_rf1.predict(x_test[independant]),
                       'feature2':model_rf2.predict(x_test[independant]),
                       'feature3':model_rf3.predict(x_test[independant]),
                       'feature4':model_rf4.predict(x_test[independant]),
                    'feature5':model_rf5.predict(x_test[independant])})
```

In [930]:

```
pred_ensemble=model_linear.predict(new_y)
pred_ensemble[id]=0
pred_ensemble[pred_ensemble>10]=10
```

In [919]:

```
min(pred_ensemble)
```

Out[919]:

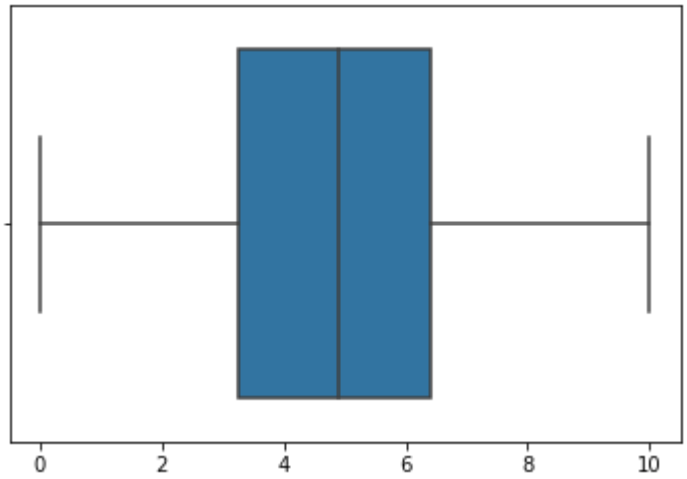
```
array([0.])
```

In [931]:

```
sns.boxplot(pred_ensemble)
```

Out[931]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bcef73be0>
```



In [852]:

```
#top features
pd.DataFrame({'variables':x_train[independant].columns,
              'importance1':model_rf1.feature_importances_,
             'importance2':model_rf2.feature_importances_}).sort_values('importa
nce',ascending=False)
```

Out[852]:

	importance	variables
5	0.592839	online_hours_mean
0	0.134841	driver_id
3	0.084861	age
1	0.034891	dayofyear
2	0.034868	dayofmonth
4	0.029832	number_of_kids
14	0.020965	dayofweek_6
13	0.016442	dayofweek_5
8	0.008257	dayofweek_0
9	0.007909	dayofweek_1
10	0.007131	dayofweek_2
6	0.006974	gender_FEMALE
7	0.006973	gender_MALE
12	0.006912	dayofweek_4
11	0.006306	dayofweek_3

In [1255]:

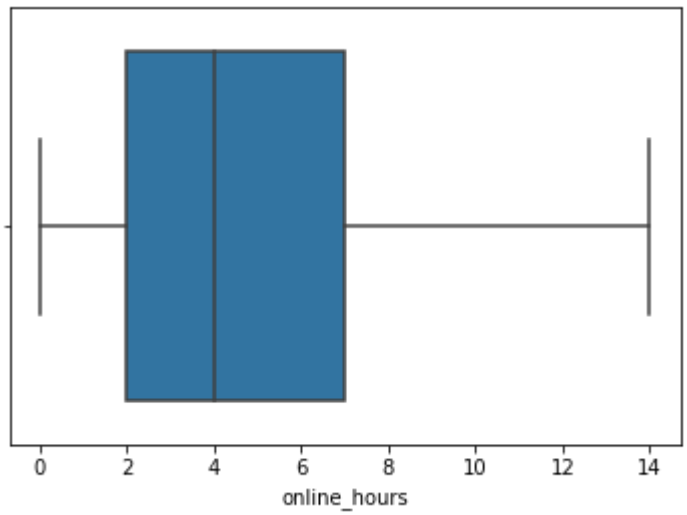
```
#predict_gbm=model_gbm.predict(x_test)
predict_rf=model_rf2.predict(x_test[independant])
predict_rf_tr=model_rf2.predict(x_train[independant])
predict_rf[id]=0#replacing the predicted value of unsenn driver id with 0
predict_rf[predict_rf>10]=10
```

In [1259]:

```
sns.boxplot(x_train.online_hours)
```

Out[1259]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1189a2048>
```



In [1256]:

```
mean_squared_error(x_test[dependant],predict_rf),mean_squared_error(x_train[depe
ndant],predict_rf_tr)
#mean_absolute_error(x_test[dependant],predict_rf)
```

Out[1256]:

```
(10.091198786724453, 0.304459254688274)
```

In [870]:

```
np.floor(predict_rf)
```

Out[870]:

```
array([2., 2., 1., ..., 0., 0., 0.])
```

In [1235]:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.tight_layout()
plt.subplot(2,2,1)
sns.boxplot(np.floor(predict_rf)).set_title("Online_hours for RF Prediction")
plt.subplot(2,2,2)
sns.boxplot(y_test).set_title("Online_hours for test_data")
plt.subplot(2,2,3)
sns.boxplot(x_train[dependant]).set_title("Online_hours for training Data")
plt.subplot(2,2,4)
sns.boxplot(pred_ensemble).set_title("Online_hours for pred ensemble")
```

Out[1235]:

```
Text(0.5,1,'Online_hours for pred ensemble')
```

In [642]:

```
driver_not_in_training
```

Out[642]:

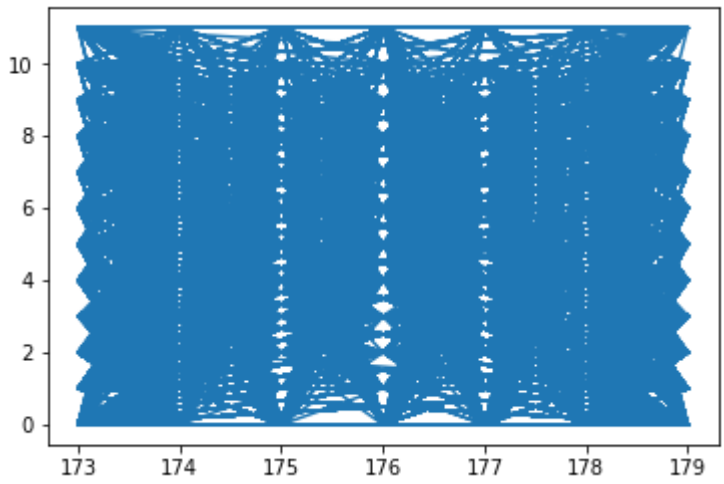
```
array([230923, 373792, 425331, 523243, 585955, 616243, 675613, 68267
8,
       743899, 756684, 772057, 808404, 854976, 934994, 971478, 99375
7,
       998740])
```

In [738]:

```
plt.plot(x_test.dayofyear,y_test)
```

Out[738]:

```
[<matplotlib.lines.Line2D at 0x1b7628af98>]
```



In [728]:

```
predict_rf[a]=np.mean(train_data.online_hours)
```