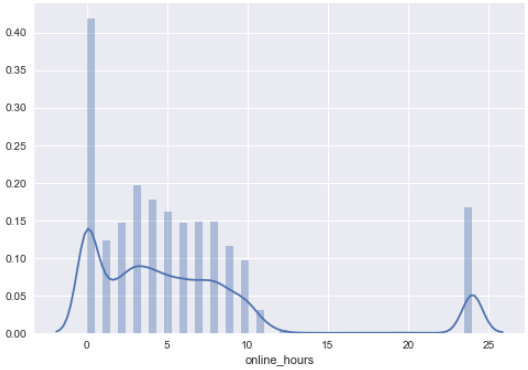
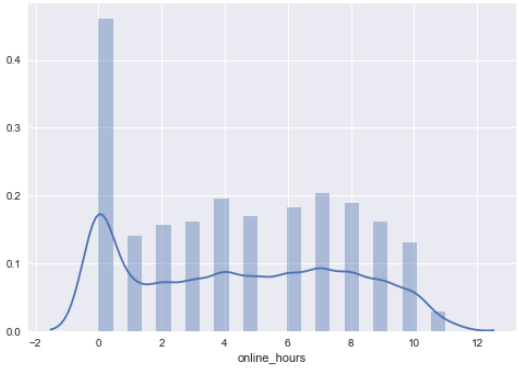
1. Data Analysis

1.1 Analysis of Online Hours for both Train and Test

Train Dataset



Test Dataset



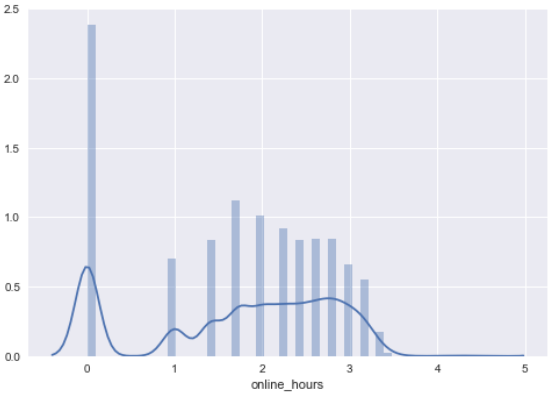
Both the graphs show that the data is non-normal

Few Other Observations:

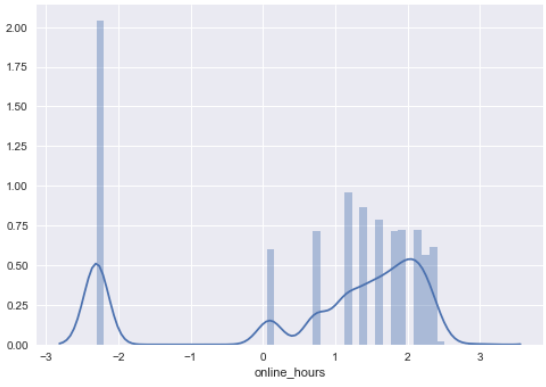
- There are days when the drivers were online for 24 hours in the Train Dataset while we do not see any such instance in Test. It will be good if we remove this from the train dataset.

- Large number of days are with zero online hours and same is the case for both train and test data

Tried few of the transformations like **Log** and **Sqrt** to turn online hours to normal and the graphs look like below:



*sns.distplot(**np.sqrt**(train['online_hours']))*



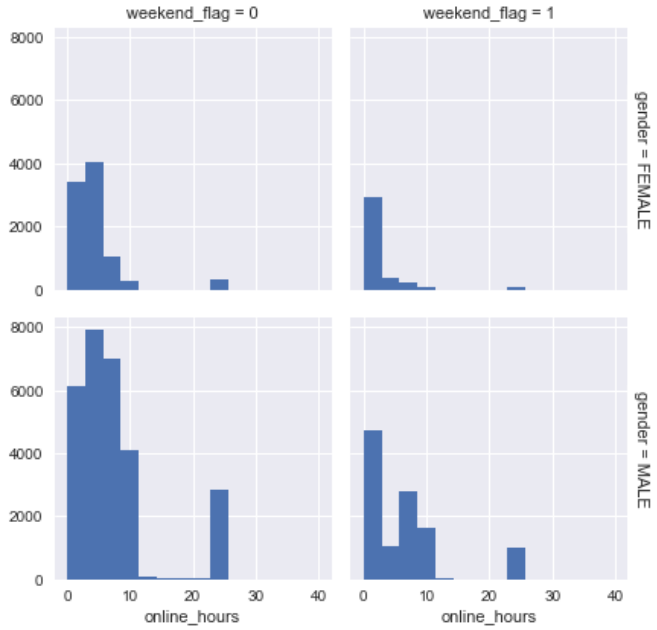
*sns.distplot(**np.log**(train['online_hours'] + 0.1))*

1.2 Analysis of various Features

1.

Weekend_Flag: (0/1)Flag to denote if the day is a weekend or not

Gender: MALE / FEMALE



Here, the above graph considers gender variable and weekend flag to show how the online hours get affected and from the graph above it seems that there are impacts during weekends.

2.

Weekday: 0 – Monday

1 – Tuesday

2 – Wednesday

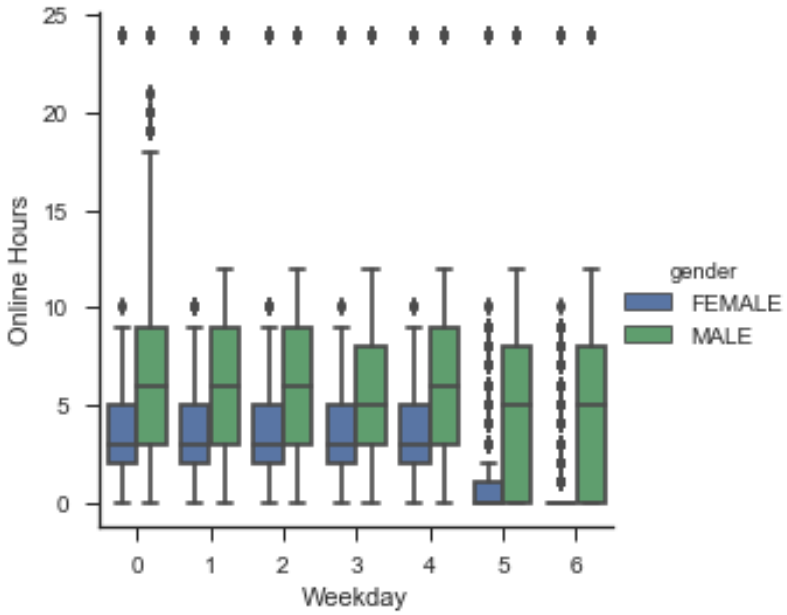
3 – Thursday

4 – Friday

5 – Saturday

6 – Sunday

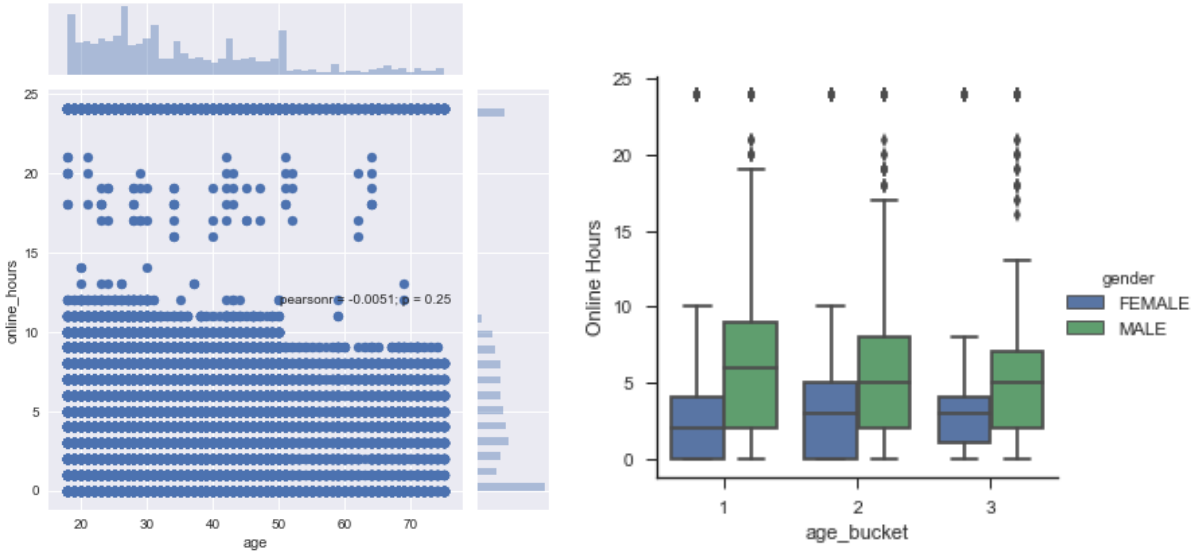
Gender: MALE / FEMALE



In the above graph, we can clearly see that the mean of few of the weekdays for male is quite different and hence weekday variable might also play a role in predicting the online hours.

3.

Age: Age of the Driver



Based on the 1st graph we can see a step like shape at 30 and 50 years. So, I went ahead and created buckets (< 30, 30-50, and > 50) and in the graph on the right we can see the mean of online hours change with these age buckets and gender.

2. Building Models

2.1 Linear Regression Model

I started with building a Linear regression model for the benchmark and below are the results.

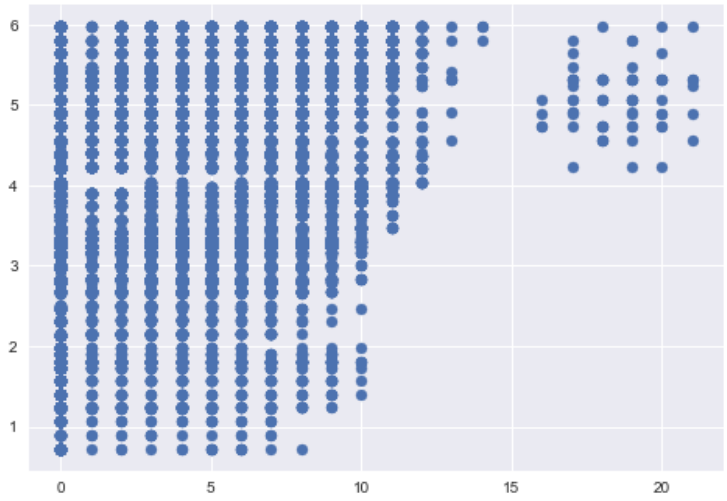
Train RMSE and Test RMSE with the pattern graph of online hours and the predicted values.

Model Details:

```
-100873064433.42949
```

	Coefficient
number_of_kids	1.669940e-01
FEMALE	-6.141208e+06
MALE	-6.141206e+06
weekend_flag_0	-1.500382e+06
weekend_flag_1	-1.500384e+06
age_bucket_1	1.008807e+11
age_bucket_2	1.008807e+11
age_bucket_3	1.008807e+11

```
<matplotlib.collections.PathCollection at 0x1270a3fd0>
RMSE: 3.1514493977991926
```



TRAIN RMSE: 3.151

TEST RMSE: 3.169

2.2 Random Forest Model

Initial List of Features Used:

'number_of_kids', 'FEMALE', 'MALE', 'weekday_0',\

'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5',\

'weekday_6', 'weekend_flag_0', 'weekend_flag_1', 'age_bucket_1',\

'age_bucket_2', 'age_bucket_3'

Model parameters:

model = RandomForestRegressor(n_estimators= 1000, max_depth=5, n_jobs=-1)

TRAIN RMSE: 3.127

TEST RMSE: 3.149

2.3 New Random Forest Model with more features

List of Features:

	cols	imp
1	Avg_kid_weekday_online_hours	0.215409
15	weekend_flag_1	0.202930
14	weekend_flag_0	0.202159
2	Avg_weekday_age_online_hours	0.187743
13	weekday_6	0.062134
5	FEMALE	0.037864
6	MALE	0.037503
12	weekday_5	0.032793
9	weekday_2	0.004802
8	weekday_1	0.004332
11	weekday_4	0.002459
7	weekday_0	0.002366
3	age	0.002140
10	weekday_3	0.001857
0	Avg_kid_age_online_hours	0.001133
17	age_bucket_2	0.001105
18	age_bucket_3	0.000871
4	number_of_kids	0.000250
16	age_bucket_1	0.000151

TRAIN RMSE: 3.09

TEST RMSE: 3.125

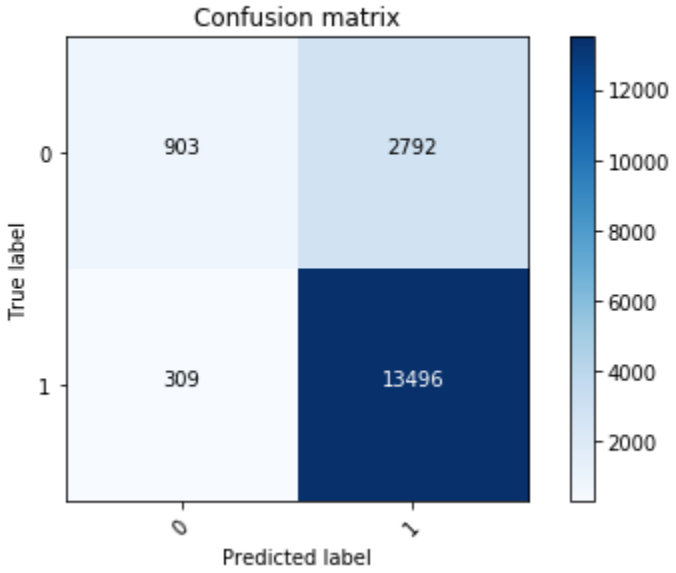
2.4 Two Step Model

- The first model will help decide whether the driver will be active/ online on a day or not

- The second model will further score the number of hours the driver will be online in case the first model says that the driver will be active

The first model was a classification model with the following accuracy

```
Accuracy of Random Forest Classifier on training data: 0.83
Accuracy of Random Forest Classifier on testing data: 0.82
```



Second Model is a regression model and the overall RMSE are as follows:

```
y_train = test['Reg_pred']
predictions = test['online_hours']
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train, predictions))
print('Mean Squared Error:', metrics.mean_squared_error(y_train, predictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_train, predictions)))
```

```
Mean Absolute Error: 2.6378858022291993
Mean Squared Error: 9.914216044883034
Root Mean Squared Error: 3.1486848119306945
```