SQL

A.

```
select A.rider_id, A.trip_id, A.trip_timestamp_utc as begintrip_timestamp_utc, A.trip_status
from
(
        select rider_id, trip_id, trip_timestamp_utc, trip_status, RANK() OVER
                (PARTITION BY rider_id ORDER BY rider_id, trip_timestamp_utc) AS Rnk
        from (select * from uber_trip
        where lower(trip_status) = "completed") B
) A
where A.Rnk = 5
```

B.

```
create table dispatch_events as
select distinct ti.trip_id as trip_id,
ti.rider_id as rider_id,
ti.driver_id as driver_id,
ti.timestamp as initiated_ts,
tca.timestamp as cancel_ts,
tco.timestamp as complete_ts
from trip_initiated ti
left join trip_cancel tca
on ti.trip_id = tca.trip_id
left join trip_complete tco
on ti.trip_id = tco.trip_id
```

C.

Select count (distinct trip_id) as initiated trips, count(distinct Case when complete_ts is not null then trip_id END) as completed_trips;

Output = No of initiated trips = No of cancelled trips + No of completed trips.

EXPERIMENTAL DESIGN:

PART A

2.1 Primary Metrics:

Supply Hours: It is equivalent to (No of cars * No of hours). It should increase because the process is supply deprived and the campaign will help overcoming the market imbalance created due to supply deprivation

Secondary Metrics:

a) **Surge**: Surge and No of drivers are inversely relational . Due to the new incentive structure the number of drivers will increase thus decreasing the surge & consumer pricing, which will eventually lead to increase in number of consumers. But after a point of time when threshold will be achieved, and supply will over pass the demand then we will have to decrease this incentive will eventually lead us to square one i.e. demand deprived situation and then we will follow this method of rolling the incentive again. This will be like a cyclic process.

b) **Driver Earnings**: Driver earnings during peak hour should also increase due to the incentive being rolled out which will eventually lead to increment in average earning of driver as well. The average earning should be more than the usual and we will have to ensure that it should sustain as well

2.2.1 We will identify the geography where supply crisis is existing. For those areas we will release this incentive-based plan to the drivers and monitor this on an hourly basis. At the end of every week evaluate and gradually deprecate the incentive structure if the market place balance has been restored.

We might also need to check if these leads to supply crisis in nearby areas. If yes, then we will have to reduce the incentive by a certain margin every week and give that in the nearby areas to prevent the market imbalance. We also need to keep in mind that consumer demand has a threshold, thus before reaching that threshold we need to ensure that there is not an excess of drivers in that area.

We also need to constantly monitor our burn and check that it is not increasing after a certain level. Thus, we need to ensure that (No of drivers * incentive) remains constant throughout. Consumer pool is mostly always constant and thus increasing burn won't have an incremental value after a certain point. Burn per incremental trip can also be monitored.

2.2.2

What type of data analysis would you perform:

We can do Analysis of Variance which is an inferential statistical method used to test the degree to which two or more groups vary or differ in an experiment. If the experiment has a great deal of variance then it indicates that there was a significant finding from the research. We can check the difference between supply before and after the incentive was rolled out. The reason for choosing this method above others is that is best suited to compare two different groups.

PART B:

Metrics of importance are listed below:

- App installs: No of app installs should increase in the nearby area where billboard has been placed
- No of active riders: No of active riders should also increase
- Trip request: Trip request should also increase due to increase in supply

Criteria for measuring the effect of campaign:

Predict what is expected and how much growth we have above the expected number. Using seasonality trend and mom growth and degrowth. If we get anything above that then it is because of our campaign success

UBER DATA CHALLENGE:

Uber's Driver Team is interested in predicting which driver signups are most likely to start driving. They have provided a sample dataset of driver signups in January 2015. The data was pulled a few months after they signed up to include the result of whether they actually completed their first trip. It also includes several pieces of background information about the driver and their vehicle.

The goal of the challenge is to understand what factors are best at predicting whether someone who signs up will actually drive, and offer suggestions to how to put those insights to use in order to help Uber. With that being said, there are 3 steps to complete:

Clean, Explore, Visualize the data as needed to find out what fraction of the signups took an initial first trip.

Build, Run, and Evaluate a Predictive Model that will help Uber determine whether or not someone who signs up will start driving. Possibly run a model selection process to discuss why the chosen model was selected, any alternatives that were considered, and any concerns that may have come up.

Discuss the Insights drawn from the model and how Uber may leverage them to get more signups to take their first trip.

Data Peek Before I take a look at the data, I will provide a brief variable description:

id: driver_id city_id: city this user signed up in signup_os: signup device of the user ("android", "ios", "website", "other") signup_channel: what channel did the driver sign up from ("offline", "paid", "organic", "referral") signup_date: timestamp of account creation; local time in the form 'YYYYMMDD' bgc_date: date of background check consent; in the form 'YYYYMMDD' vehicle_added_date: date when driver's vehicle information was uploaded; in the form 'YYYYMMDD' first_completed_date: date of the first trip as a driver; in the form 'YYYYMMDD' vehicle_make: make of vehicle uploaded (i.e. Honda, Ford, Kia) vehicle_model: model of vehicle uploaded (i.e. Accord, Prius, 350z) vehicle_year: year that the car was made; in the form 'YYYY'

---------- Data Exploration ----------

In [3]:
```
#Importing libraries for computation:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
#Reading the file and storing in a dataframe:
df = pd.read_csv('C:/Users/abhip/OneDrive/Desktop/Uber/Uber_prod.csv')
df.replace('', np.nan, inplace=True)
```

In [5]:
```
# Having an intial look at the data to determine what kind of data clean
ing is required:
df.head(3)
```

Out[5]:

	id	city_name	signup_os	signup_channel	signup_timestamp	bgc_date
0	082befb0-c1de-4c14-8700-94a7943a7545	Strark	NaN	R2D	2017-07-06T20:42:17Z	NaN
1	1ae6156f-63fc-40cf-9734-0995978c4b6e	Berton	ios web	Dost	2017-07-03T17:41:07Z	2017-07-03T17:42:06Z
2	2a4a4eef-14ef-4ceb-82eb-66f1f7d0d219	Berton	NaN	R2D	2017-07-10T22:55:29Z	NaN

In [6]:
```
# Checking the shape of the dataset:
df.shape #We can see that there are 12357 rows and 11 columns or variabl
es.
```

Out[6]:
```
(12357, 11)
```

In [7]:
```
#Checking names of all the columns:
df.columns
```

Out[7]:
```
Index(['id', 'city_name', 'signup_os', 'signup_channel', 'signup_timest
amp',
       'bgc_date', 'vehicle_added_date', 'vehicle_make', 'vehicle_mode
l',
       'vehicle_year', 'first_completed_trip_timestamp'],
      dtype='object')
```

In [8]:
```
#Checking for column datatype and non-missing values:
df.info() #I can also see that most of the columns are of object type so
 we will need to change them to correct datatype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12357 entries, 0 to 12356
Data columns (total 11 columns):
id                                12357 non-null object
city_name                         12357 non-null object
signup_os                         6953 non-null object
signup_channel                    11143 non-null object
signup_timestamp                  11194 non-null object
bgc_date                          7803 non-null object
vehicle_added_date                11121 non-null object
vehicle_make                      11727 non-null object
vehicle_model                     12357 non-null object
vehicle_year                      12357 non-null int64
first_completed_trip_timestamp    6790 non-null object
dtypes: int64(1), object(10)
memory usage: 1.0+ MB
```

In [9]:
```
#Just adding the parse argument we can change the date columns to the co
rrect datatype:
df = pd.read_csv('C:/Users/abhip/OneDrive/Desktop/Uber/Uber_prod.csv',pa
rse_dates=[4,5,6,10])
```

In [10]:
```
#Checking for the datatype again after tansformation:
df.info() #We can see that
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12357 entries, 0 to 12356
Data columns (total 11 columns):
id                                12357 non-null object
city_name                         12357 non-null object
signup_os                         6953 non-null object
signup_channel                    11143 non-null object
signup_timestamp                  11194 non-null datetime64[ns, UTC]
bgc_date                          7803 non-null datetime64[ns, UTC]
vehicle_added_date                11121 non-null datetime64[ns, UTC]
vehicle_make                      11727 non-null object
vehicle_model                     12357 non-null object
vehicle_year                      12357 non-null int64
first_completed_trip_timestamp    6790 non-null datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](4), int64(1), object(6)
memory usage: 1.0+ MB
```

In [11]:
```
#We can see that a lot of columns have missing values. Let us create a t
able for the same to have a better view:
# Function to calculate missing values by column:
def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * mis_val / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Your selected dataframe has " + str(df.shape[1]) + " col
umns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")

        # Return the dataframe with missing information
        return mis_val_table_ren_columns
```

In [12]:
```
# Generating a summary of missing values:
missing_values_table(df)
```

```
Your selected dataframe has 11 columns.
There are 7 columns that have missing values.
```

Out[12]:

	Missing Values	% of Total Values
first_completed_trip_timestamp	5567	45.1
signup_os	5404	43.7
bgc_date	4554	36.9
vehicle_added_date	1236	10.0
signup_channel	1214	9.8
signup_timestamp	1163	9.4
vehicle_make	630	5.1

In [13]:
```
#Removing the columns in which all columns named signup_timestamp, bgc_d
ate, vehicle_added_date & first_completed_trip_timestamp have null value
s:
df=df.dropna(subset=['signup_timestamp', 'bgc_date', 'vehicle_added_dat
e', 'first_completed_trip_timestamp'], how = 'all')
df.shape
```

Out[13]: (12329, 11)

In [14]:
```
#Replacing NA values in signup_os by other values:
df['signup_os'].fillna('other', inplace=True)
df['signup_os'].value_counts(dropna = False)
```

Out[14]:
```
other          6188
ios web        3227
android web    1850
mac             717
windows         347
Name: signup_os, dtype: int64
```

In [15]:
```
#Replacing NA values in signup_channel by other values:
df['signup_channel'].fillna('other', inplace=True)
df['signup_channel'].value_counts(dropna = False)
```

Out[15]:
```
Referral    5750
R2D         2391
Paid        1703
other       1208
Organic     1189
Dost          88
Name: signup_channel, dtype: int64
```

In [16]:
```
#We can see that there are almost 100 types of car variants present so i
n order to not clutter data we will categorize any vehicle with count le
ss than 200 under Other category:
#Making a new count column:
df['vehicle_make_count'] = df['vehicle_make'].groupby(df['vehicle_make'
]).transform('count')
#Transforming the vehicle_make variable with required values:
df["vehicle_make"] = np.where(df['vehicle_make_count']<= 200, 'other', d
f['vehicle_make'])

df['vehicle_make'].fillna('other', inplace=True)
```

In [17]:
```
# Verifying if the transformation in vehicle_make was correct or not:
df.head(30)
```

Out[17]:

	id	city_name	signup_os	signup_channel	signup_timestamp	bgc_c
0	082befb0-c1de-4c14-8700-94a7943a7545	Strark	other	R2D	2017-07-06 20:42:17+00:00	NaT
1	1ae6156f-63fc-40cf-9734-0995978c4b6e	Berton	ios web	Dost	2017-07-03 17:41:07+00:00	2017-07-03 17:42:06+0
2	2a4a4eef-14ef-4ceb-82eb-66f1f7d0d219	Berton	other	R2D	2017-07-10 22:55:29+00:00	NaT
3	56fe7597-3ad8-4798-8be8-5fbc4e2d3151	Berton	ios web	Referral	2017-07-27 18:27:21+00:00	2017-07-27 18:31:43+0
4	67370341-68a5-415f-acf2-be58832a8f9c	Wrouver	other	Referral	2017-07-17 22:20:35+00:00	2017-07-17 22:21:09+0
5	6baded87-3526-4cad-9745-1d48e7e451ec	Berton	ios web	Referral	2017-07-11 23:01:39+00:00	2017-07-11 23:02:41+0
6	6d5f9955-76fd-4795-b115-676c5265b767	Berton	ios web	Referral	2017-07-13 23:51:56+00:00	2017-07-13 23:52:52+0
7	8348227f-7275-419b-85c1-687828fab699	Berton	other	R2D	2017-07-28 06:10:22+00:00	NaT
8	88c42231-e6d4-42fb-90e7-acb111a9162f	Wrouver	android web	Referral	2017-07-08 02:06:13+00:00	2017-07-08 02:07:34+0
9	e19c8ec7-6b4e-4b27-9ea0-c8c497784aa8	Strark	android web	other	2017-07-22 03:00:51+00:00	2017-07-22 03:03:49+0

	id	city_name	signup_os	signup_channel	signup_timestamp	bgc_c
10	f859ccbd-7922-43ce-a2aa-a1da75d72a71	Strark	other	R2D	NaT	NaT
11	fd81cab6-0e81-4678-8615-589cd64d49e3	Strark	ios web	Referral	2017-07-10 21:52:32+00:00	2017-07-10 21:53:22+0
12	2f402126-ddde-4e70-ae4b-e94802145d57	Strark	windows	Referral	2017-07-26 18:55:36+00:00	2017-07-26 18:57:50+0
13	521f5a45-e4b1-4f4d-9088-bdcfbc8728fc	Berton	other	R2D	NaT	NaT
14	792ace92-a621-4c08-b742-94a4dc094efa	Berton	ios web	Organic	2017-07-21 19:00:16+00:00	2017-07-21 19:01:51+0
15	79de2f24-2ccb-4741-8313-b8d869bab051	Strark	android web	other	2017-07-06 04:52:36+00:00	2017-07-06 04:53:33+0
16	ae406683-0ea9-4bce-82be-b6ded4779a38	Strark	other	Referral	2017-07-24 19:33:54+00:00	NaT
17	ae406683-0ea9-4bce-82be-b6ded4779a38	Strark	other	Referral	2017-07-24 19:33:54+00:00	NaT
18	b9433ce9-2562-470e-938a-4bab8102ceb2	Berton	ios web	Dost	NaT	2017-07-09 02:23:42+0
19	cbaf00b2-4d7f-4b0e-8162-5437e49363ba	Strark	android web	Referral	2017-07-08 23:47:31+00:00	2017-07-08 23:48:40+0

	id	city_name	signup_os	signup_channel	signup_timestamp	bgc_c
20	e0513d05-942c-47d4-8102-2fff5d5307b1	Strark	mac	Organic	2017-07-25 20:41:45+00:00	2017-07-25 20:42:21+0
21	e0513d05-942c-47d4-8102-2fff5d5307b1	Strark	mac	other	2017-07-25 20:41:45+00:00	2017-07-25 20:42:21+0
22	e0513d05-942c-47d4-8102-2fff5d5307b1	Strark	mac	Organic	2017-07-25 20:41:45+00:00	2017-07-25 20:42:21+0
23	e0513d05-942c-47d4-8102-2fff5d5307b1	Strark	mac	Organic	2017-07-25 20:41:45+00:00	2017-07-25 20:42:21+0
24	1539f166-a783-44d7-9e7d-ecf48d98b84b	Strark	other	other	2017-07-31 00:33:44+00:00	2017-07-31 00:34:52+0
25	2f150648-07f8-4df8-9da2-4f58e22b9897	Berton	other	Referral	2017-07-31 02:05:45+00:00	2017-07-31 02:06:56+0
26	2f4edcf0-177e-48a6-8084-ac7a6665152c	Strark	android web	Referral	2017-07-13 04:35:12+00:00	2017-07-13 04:35:45+0
27	83379aed-0c39-4580-aac5-bb10b53066e0	Berton	other	R2D	2017-07-25 00:23:46+00:00	NaT
28	8598e5a7-af85-4d55-9b4c-2e02e7f3cb0d	Strark	other	other	2017-07-20 19:33:52+00:00	NaT
29	8b36e80a-926d-4384-8090-864e412bbdb1	Strark	ios web	Referral	2017-07-23 23:58:39+00:00	2017-07-23 23:59:07+0

In [18]:
```
# Iterate through the columns and extract only dates from the datetime c
olumns:
for col in list(df.columns):
    # Select columns that should be date:
    if ('signup_timestamp' in col or 'bgc_date' in col or 'vehicle_added
_date' in col or 'first_completed_trip_timestamp' in
        col):
        # Convert the data type to float
        df[col] = df[col].dt.date
```

In [19]:
```
df.info() #We can see required date columns having the right object typ
e:
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12329 entries, 0 to 12356
Data columns (total 12 columns):
id                                  12329 non-null object
city_name                           12329 non-null object
signup_os                           12329 non-null object
signup_channel                      12329 non-null object
signup_timestamp                    11194 non-null object
bgc_date                            7803 non-null object
vehicle_added_date                  11121 non-null object
vehicle_make                        12329 non-null object
vehicle_model                       12329 non-null object
vehicle_year                        12329 non-null int64
first_completed_trip_timestamp      6790 non-null object
vehicle_make_count                  11700 non-null float64
dtypes: float64(1), int64(1), object(10)
memory usage: 1.5+ MB
```

In [20]:
```
#Adding 2 new columns with difference in dates to show that the dates in
 3 columns are equal to each other
df['1_Diff'] = (df['signup_timestamp'] - df['bgc_date']).dt.days #Difefr
ence between signup_timestamp & bgc_date
df['2_Diff'] = (df['signup_timestamp'] - df['vehicle_added_date']).dt.da
ys ##Difefrence between signup_timestamp & vehicle_added_dat
df['3_Diff'] = (df['bgc_date'] - df['vehicle_added_date']).dt.days
df.head(3)
```

Out[20]:

	id	city_name	signup_os	signup_channel	signup_timestamp	bgc_date	ve
0	082befb0-c1de-4c14-8700-94a7943a7545	Strark	other	R2D	2017-07-06	NaT	20
1	1ae6156f-63fc-40cf-9734-0995978c4b6e	Berton	ios web	Dost	2017-07-03	2017-07-03	20
2	2a4a4eef-14ef-4ceb-82eb-66f1f7d0d219	Berton	other	R2D	2017-07-10	NaT	20

In [21]:
```
df['2_Diff'].value_counts(dropna = True) # we can see that almost 80% of
 the values in vehicle_added_date and signup time column are same thus r
eplacing all the null occurences in signup  with vehicle_added_date is a
 good option
```

Out[21]:

0.0	9823
-1.0	75
-2.0	22
-3.0	16
-7.0	13
-4.0	13
-6.0	10
-5.0	10
-32.0	6
-12.0	5
-8.0	5
-16.0	5
-15.0	4
-18.0	4
-11.0	3
-28.0	3
-31.0	3
-29.0	3
-9.0	3
-21.0	3
-37.0	3
-23.0	3
-20.0	2
-13.0	2
-24.0	2
-10.0	2
-17.0	2
-47.0	2
-85.0	2
-22.0	2
-39.0	2
-42.0	1
-27.0	1
17.0	1
-34.0	1
3.0	1
-65.0	1
-88.0	1
-44.0	1
-56.0	1
-68.0	1
-14.0	1
24.0	1
-38.0	1
-66.0	1
-60.0	1
-67.0	1
-57.0	1
-35.0	1
-84.0	1
-53.0	1
-64.0	1
-19.0	1
-49.0	1
-59.0	1
-69.0	1
-33.0	1

```
 1.0        1
-71.0       1
Name: 2_Diff, dtype: int64
```

In [22]:
```
#Filling the signup timestamp column:
df[['signup_timestamp']].apply(lambda x: x.fillna(value=df['vehicle_adde
d_date']))
```

Out[22]:

	signup_timestamp
0	2017-07-06
1	2017-07-03
2	2017-07-10
3	2017-07-27
4	2017-07-17
5	2017-07-11
6	2017-07-13
7	2017-07-28
8	2017-07-08
9	2017-07-22
10	2017-07-30
11	2017-07-10
12	2017-07-26
13	2017-07-31
14	2017-07-21
15	2017-07-06
16	2017-07-24
17	2017-07-24
18	2017-07-09
19	2017-07-08
20	2017-07-25
21	2017-07-25
22	2017-07-25
23	2017-07-25
24	2017-07-31
25	2017-07-31
26	2017-07-13
27	2017-07-25
28	2017-07-20
29	2017-07-23
...	...
12327	2017-07-23

	signup_timestamp
12328	2017-07-23
12329	2017-07-09
12330	2017-07-21
12331	2017-07-28
12332	2017-07-11
12333	2017-07-02
12334	2017-07-02
12335	2017-07-10
12336	2017-07-10
12337	2017-07-07
12338	2017-07-13
12339	2017-07-14
12340	2017-07-11
12341	2017-07-24
12342	2017-07-31
12343	2017-07-06
12344	2017-07-05
12345	2017-07-13
12346	2017-07-31
12347	2017-07-31
12348	2017-07-30
12349	2017-07-12
12350	2017-07-13
12351	2017-07-04
12352	2017-07-07
12353	2017-07-09
12354	2017-07-09
12355	2017-07-17
12356	2017-07-07

12329 rows × 1 columns

In [23]:
```
#Checking for value counts
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12329 entries, 0 to 12356
Data columns (total 15 columns):
id                                12329 non-null object
city_name                         12329 non-null object
signup_os                         12329 non-null object
signup_channel                    12329 non-null object
signup_timestamp                  11194 non-null object
bgc_date                          7803 non-null object
vehicle_added_date                11121 non-null object
vehicle_make                      12329 non-null object
vehicle_model                     12329 non-null object
vehicle_year                      12329 non-null int64
first_completed_trip_timestamp    6790 non-null object
vehicle_make_count                11700 non-null float64
1_Diff                            7094 non-null float64
2_Diff                            10081 non-null float64
3_Diff                            7020 non-null float64
dtypes: float64(4), int64(1), object(10)
memory usage: 1.8+ MB
```

In [24]:
```
df=df.dropna(subset=['signup_timestamp'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11194 entries, 0 to 12356
Data columns (total 15 columns):
id                                11194 non-null object
city_name                         11194 non-null object
signup_os                         11194 non-null object
signup_channel                    11194 non-null object
signup_timestamp                  11194 non-null object
bgc_date                          7094 non-null object
vehicle_added_date                10081 non-null object
vehicle_make                      11194 non-null object
vehicle_model                     11194 non-null object
vehicle_year                      11194 non-null int64
first_completed_trip_timestamp    6149 non-null object
vehicle_make_count                10619 non-null float64
1_Diff                            7094 non-null float64
2_Diff                            10081 non-null float64
3_Diff                            6382 non-null float64
dtypes: float64(4), int64(1), object(10)
memory usage: 1.4+ MB
```

In [91]:
```
#Making a new column with a value of 1 if diff[first_completed_trip_time
stamp  - signup_timestamp ]<=30 days and 0 otherwise:
df['4_Diff'] = (df['first_completed_trip_timestamp'] - df['signup_timest
amp'] ).dt.days

df["dep_var"] = np.where(df['4_Diff']<=30, 1, 0)
ct = df.groupby('dep_var')['id'].count()
ct
```

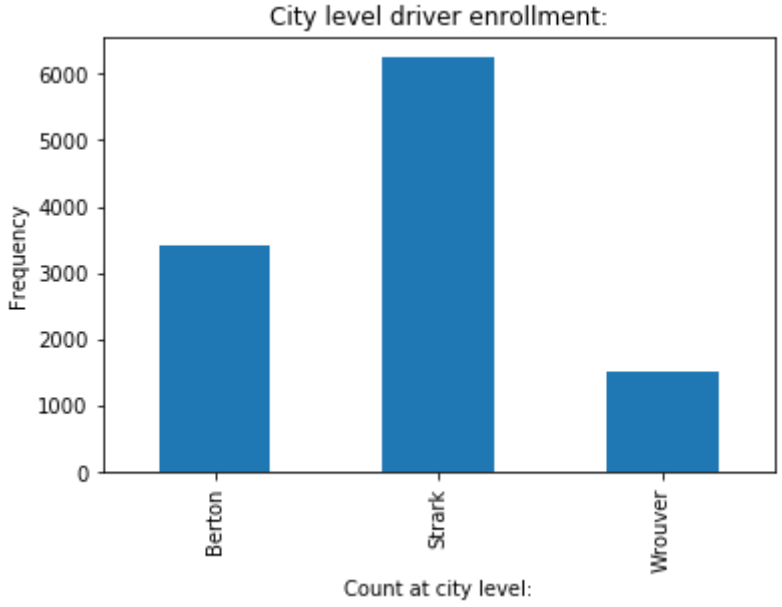
Out[91]:
```
dep_var
0    5045
1    6149
Name: id, dtype: int64
```

We can see that 6149 out of 12357 i.e. 49.7% people will drive with Uber within a month.

In [26]:
```
#Selecting final columns for model prediction:
df_fin = df.loc[:, ['city_name', 'signup_os', 'signup_channel','signup_t
imestamp', 'vehicle_make', 'vehicle_year', 'dep_var']]
df_fin.info(10)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11194 entries, 0 to 12356
Data columns (total 7 columns):
city_name           11194 non-null object
signup_os           11194 non-null object
signup_channel      11194 non-null object
signup_timestamp    11194 non-null object
vehicle_make        11194 non-null object
vehicle_year        11194 non-null int64
dep_var             11194 non-null int32
dtypes: int32(1), int64(1), object(5)
memory usage: 655.9+ KB
```

In [27]:
```
# Let us explore a relation of all the independent variables separately
 and also against dependent variable and see if there is any insigth in
 that:
# 1. City against dep_var:

count_default_payments = pd.value_counts(df['city_name'], sort = True).
sort_index()
count_default_payments.plot(kind = 'bar')
plt.title("City level driver enrollment:")
plt.xlabel("Count at city level:")
plt.ylabel("Frequency")
plt.show()

# We can see that the largest number of drivers are from city "Strark"
```

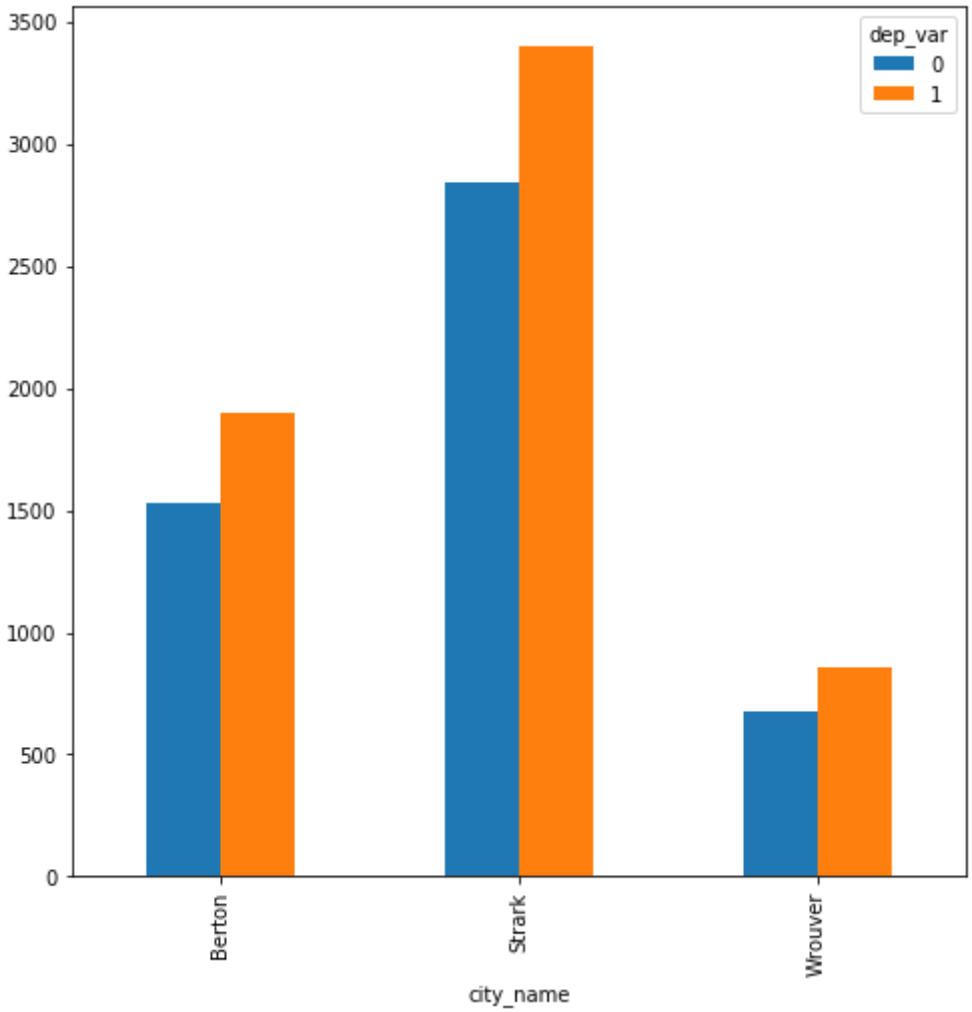


In [28]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.city_name])

clarity_color_table
```

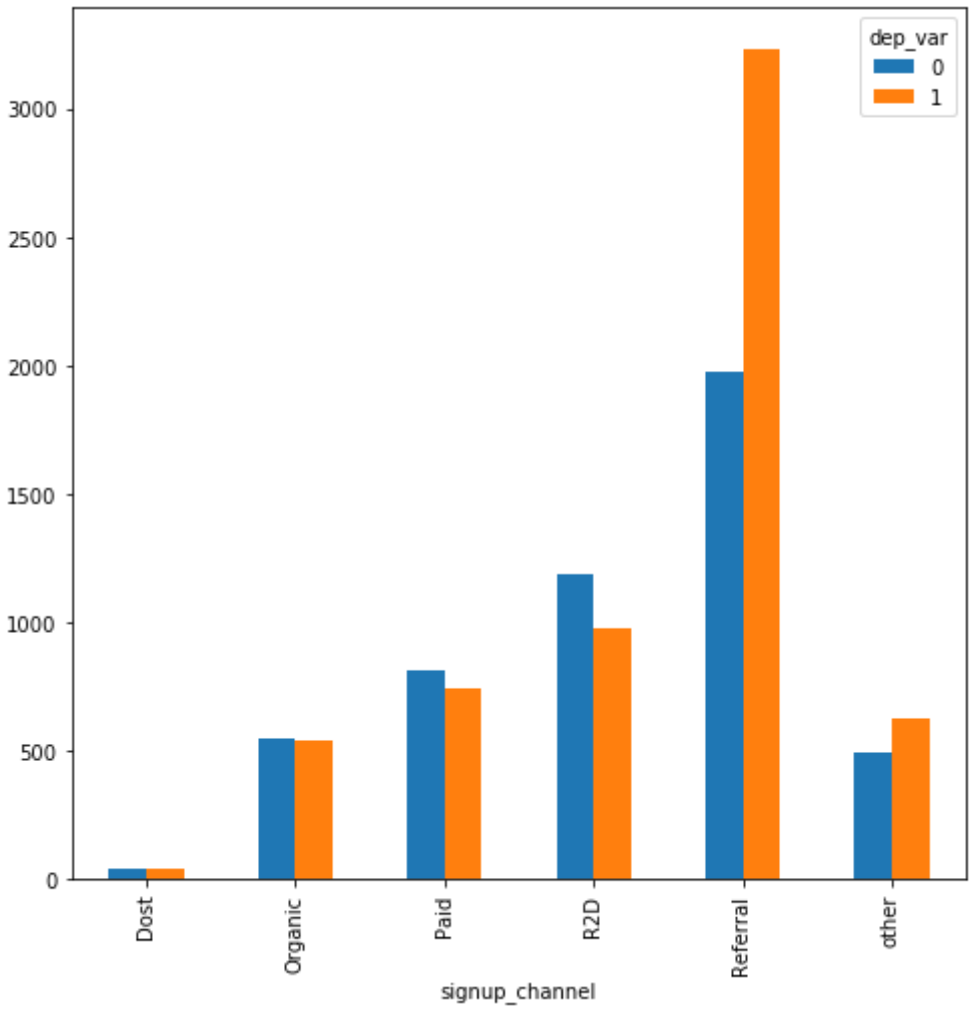
Out[28]:

dep_var	0	1
city_name		
Berton	1526	1896
Strark	2846	3398
Wrouver	673	855

In [29]:
```
clarity_color_table.plot(kind="bar",
                         figsize=(8,8),
                         stacked=False)
plt.show()
```



In [30]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.signup_channel])

clarity_color_table.plot(kind="bar",
                 figsize=(8,8),
                 stacked=False)
plt.show()
```

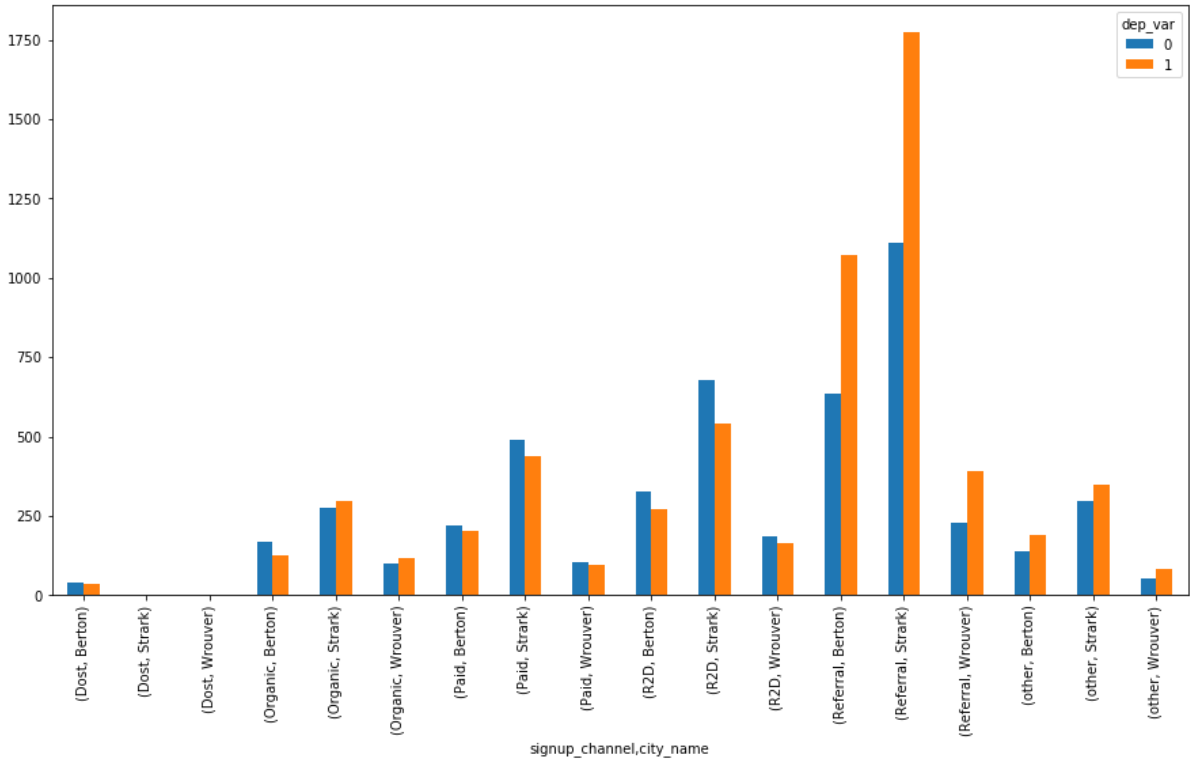


In [31]:
```
#col_names = ['city_name', 'signup_os', 'signup_channel', 'vehicle_mak
e', 'vehicle_year']
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.signup_channel, df_fin.city_nam
e])

clarity_color_table
```

Out[31]:

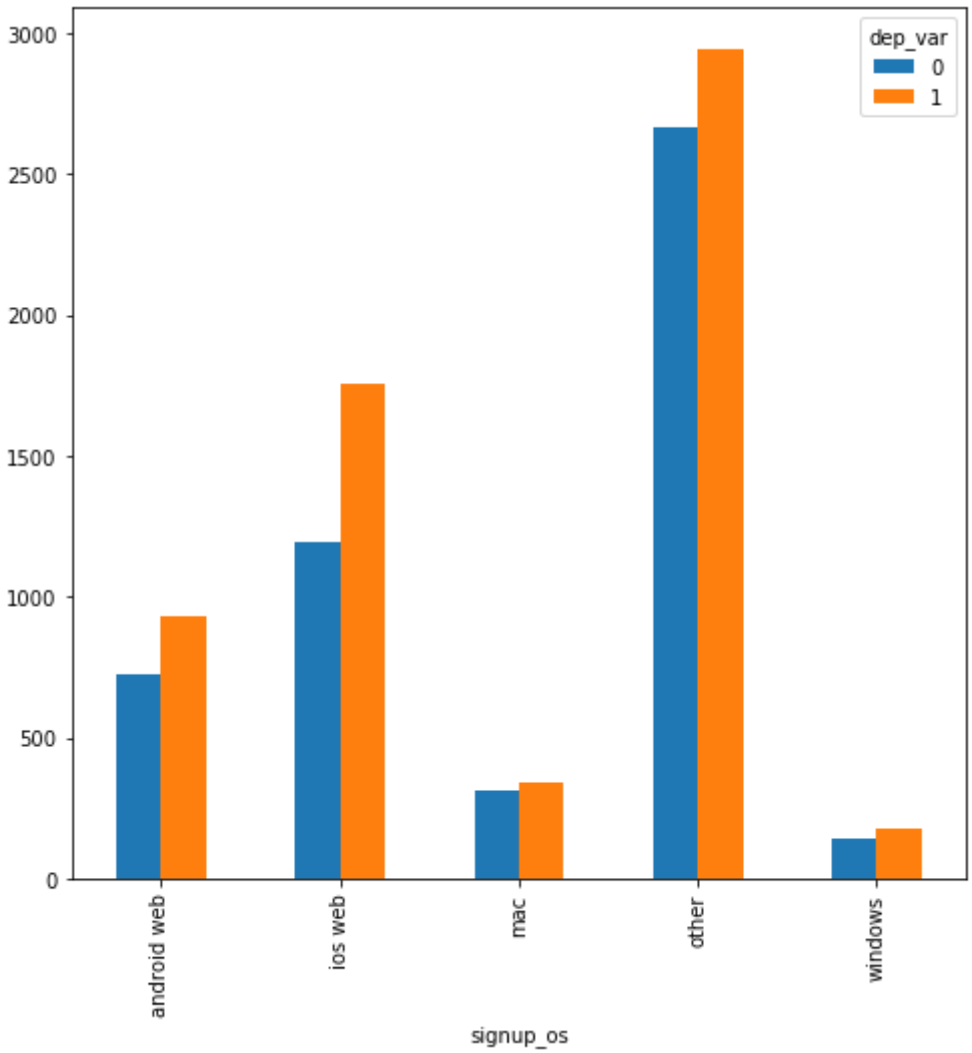
	dep_var	0	1
signup_channel	**city_name**		
Dost	**Berton**	39	35
	Strark	1	0
	Wrouver	1	0
Organic	**Berton**	169	127
	Strark	274	295
	Wrouver	100	117
Paid	**Berton**	220	202
	Strark	489	440
	Wrouver	104	97
R2D	**Berton**	325	270
	Strark	676	543
	Wrouver	187	166
Referral	**Berton**	635	1072
	Strark	1110	1773
	Wrouver	227	390
other	**Berton**	138	190
	Strark	296	347
	Wrouver	54	85

In [32]:
```
clarity_color_table.plot(kind="bar",
                         figsize=(15,8),
                         stacked=False)
plt.show()
```

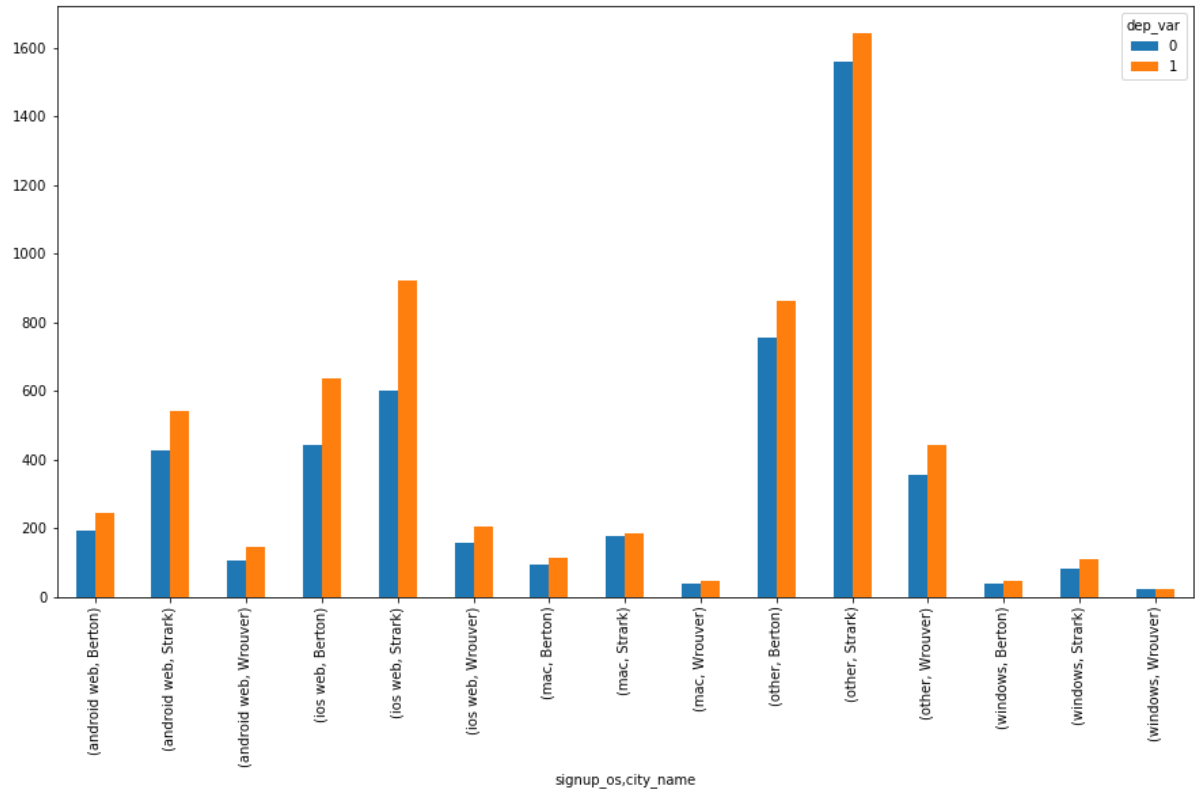


In [33]:
```
#From the above we can see that Referral channel is the best channel & D
ost is the worst channel for overall driver count
# We can also see that R2D & Paid channel have potential for growth so w
e can probably think of shifting our focus in terms of budgeting and oth
er efforts from dost to these 2 channels more:
# Wrouver city has the lowest share in the overall drivers turnout to dr
ive with Uber
#Strark has the best turout from all the channels with best from Referra
l channel
```

In [34]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                                  index= [df_fin.signup_os])

clarity_color_table.plot(kind="bar",
                         figsize=(8,8),
                         stacked=False)
plt.show()
```

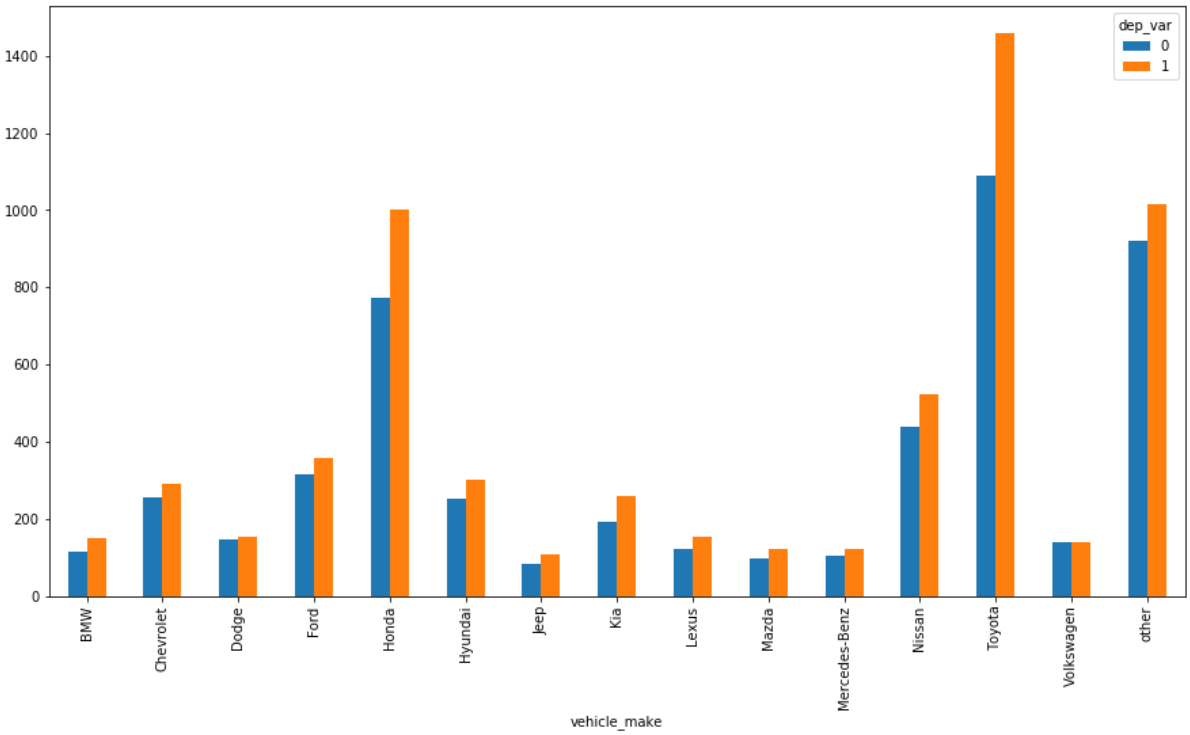


In [35]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.signup_os, df_fin.city_name])

clarity_color_table.plot(kind="bar",
                 figsize=(15,8),
                 stacked=False)
plt.show()
```



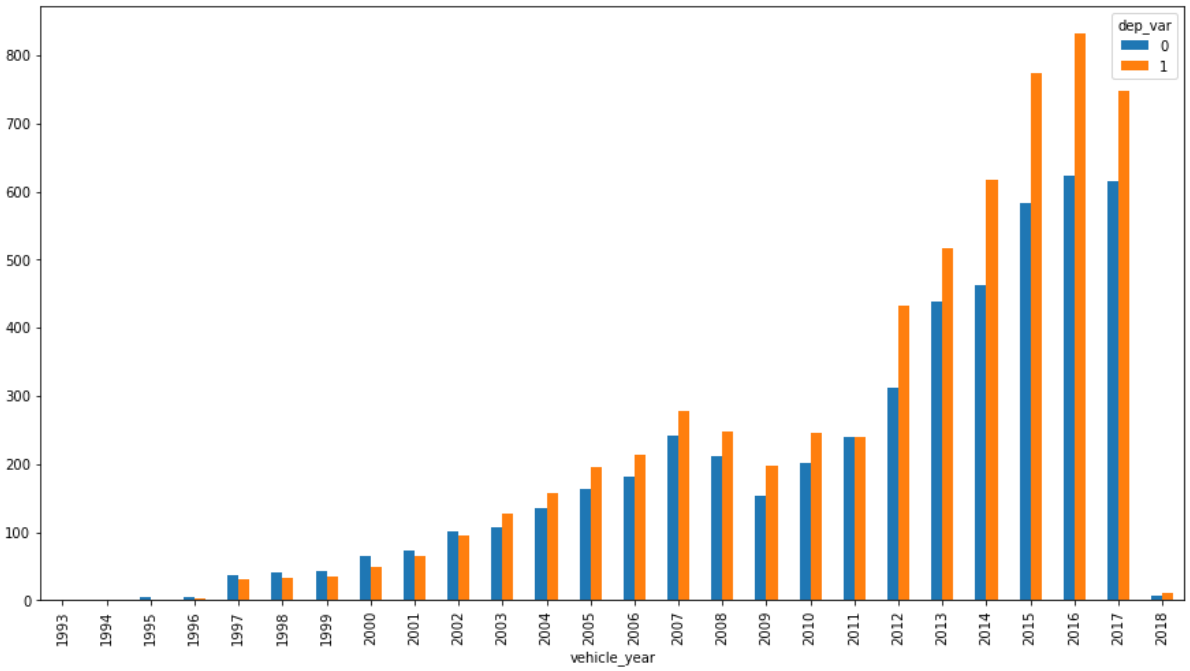
In [36]:
```
#We can see that the other category is the highest when categorized unde
r signup_os drivers fall into. This shows that the team responsible for
 taking a note of signup_os or people filling automatic forms are exclud
ing this particular information. We should make it mandatory to get this
 field filled
# Windows and Mac have the lowest share showing that most of the drivers
 coming to register do it more on their mobiles given the other category
 fall in the same weightage as right now if we exclude otehr category. W
e can probbaly make a budget cut on these 2 channels and focus more on m
obile platforms
#Ios appears to get the most customers and thus we can focus more on thi
s channel in future if we need instant growth
```

In [37]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.vehicle_make])

clarity_color_table.plot(kind="bar",
                 figsize=(15,8),
                 stacked=False)
plt.show()
```



In [38]:
```
#We can  see a clear trend that Honda & Toyota brands are more likely to
 make it through and driver for Uber so we can probbaly focus more on
#-- Sedan type and we can also check the pricing model for Sedan and try
 to implement other types models also in similar way..
# Nissan also looks like a potential brand on which we can put more focu
s in future
```

In [39]:
```
clarity_color_table = pd.crosstab(columns=df_fin.dep_var,
                          index= [df_fin.vehicle_year])

clarity_color_table.plot(kind="bar",
                 figsize=(15,8),
                 stacked=False)
plt.show()
```



In [40]:
```
#We can see that probability of vehicles to ride with Uber is clearly de
pending on vehicle_make year and vehicles which were made before 2012 we
re less likely to onboard
# We can probably check if these cars with vehicle date before 2012 had
 issues with papers or some specific problem like pollution level from t
he cars and design a strategy for future.
```

In [41]:
```
#Transforming columns to perform modelling:

#Transforming signup_timestamp column by sorting it in ascending aorder
 and providing an integer value against each date:
df_fin.sort_values('signup_timestamp')
```

Out[41]:

	city_name	signup_os	signup_channel	signup_timestamp	vehicle_make	vehicle_
9486	Strark	ios web	other	2017-07-01	other	2015
326	Strark	ios web	Referral	2017-07-01	Honda	1996
2054	Strark	other	other	2017-07-01	Nissan	2007
907	Strark	other	Organic	2017-07-01	Chevrolet	2011
11121	Wrouver	android web	other	2017-07-01	other	2004
11130	Berton	android web	Referral	2017-07-01	Ford	2002
4943	Berton	ios web	Referral	2017-07-01	Toyota	2014
8807	Strark	mac	Referral	2017-07-01	Hyundai	2014
8784	Wrouver	other	Paid	2017-07-01	other	2011
8777	Wrouver	ios web	Referral	2017-07-01	Chevrolet	2008
2037	Wrouver	other	Referral	2017-07-01	Toyota	2013
11148	Berton	mac	Referral	2017-07-01	Honda	2016
4970	Strark	mac	Organic	2017-07-01	Ford	2012
11172	Wrouver	mac	Paid	2017-07-01	Toyota	2009
2024	Strark	windows	Referral	2017-07-01	Nissan	2015
5034	Strark	android web	Paid	2017-07-01	other	2015
8723	Wrouver	other	R2D	2017-07-01	Toyota	1999
5080	Strark	other	Referral	2017-07-01	Ford	2005
2011	Strark	android web	Referral	2017-07-01	Honda	2013
8655	Strark	ios web	Referral	2017-07-01	other	2013
8930	Strark	android web	Referral	2017-07-01	Nissan	2017
3455	Strark	other	Referral	2017-07-01	Mazda	2007
4890	Wrouver	ios web	Paid	2017-07-01	Toyota	2006
2076	Berton	ios web	Referral	2017-07-01	Toyota	2016
3715	Strark	other	Referral	2017-07-01	BMW	2015
3710	Berton	other	Referral	2017-07-01	other	2015
10958	Berton	ios web	Referral	2017-07-01	Ford	2017
9212	Strark	other	other	2017-07-01	Dodge	2015

	city_name	signup_os	signup_channel	signup_timestamp	vehicle_make	vehicle_
4787	Strark	ios web	Referral	2017-07-01	Toyota	2000
4800	Berton	other	Referral	2017-07-01	Toyota	2007
...	...	...	...	...	...	...
1763	Strark	other	R2D	2017-07-31	Dodge	2014
1766	Strark	other	R2D	2017-07-31	Toyota	2007
246	Berton	other	Referral	2017-07-31	Toyota	2014
5802	Berton	other	Paid	2017-07-31	Honda	2015
1796	Berton	other	R2D	2017-07-31	Chevrolet	2007
11322	Strark	other	R2D	2017-07-31	Toyota	2008
6352	Strark	other	Referral	2017-07-31	Ford	2017
11321	Berton	other	R2D	2017-07-31	Mazda	2016
11304	Strark	other	other	2017-07-31	other	2017
11217	Strark	other	Paid	2017-07-31	Kia	2017
6343	Berton	other	Organic	2017-07-31	Nissan	2013
6340	Strark	other	other	2017-07-31	Nissan	2016
11226	Wrouver	other	Referral	2017-07-31	Toyota	2016
1656	Strark	other	Referral	2017-07-31	Toyota	2016
1657	Strark	other	Organic	2017-07-31	Chevrolet	2017
6294	Berton	other	Referral	2017-07-31	Chevrolet	2017
11242	Berton	other	Referral	2017-07-31	Honda	2013
6250	Strark	other	other	2017-07-31	Volkswagen	2000
11255	Berton	other	Referral	2017-07-31	Hyundai	2012
6228	Strark	other	Referral	2017-07-31	Honda	2017
6204	Wrouver	other	Referral	2017-07-31	Toyota	2002
1677	Strark	other	Paid	2017-07-31	Ford	1997
6194	Wrouver	other	R2D	2017-07-31	Volkswagen	2016
11282	Strark	other	R2D	2017-07-31	Ford	2007
11283	Strark	other	Referral	2017-07-31	Chevrolet	2017
1694	Strark	other	Organic	2017-07-31	other	2012
6137	Strark	other	Paid	2017-07-31	Dodge	2007
11303	Strark	other	Referral	2017-07-31	other	2017
11312	Berton	other	Paid	2017-07-31	other	2014

	city_name	signup_os	signup_channel	signup_timestamp	vehicle_make	vehicle_
6182	Strark	other	R2D	2017-07-31	Toyota	2014

11194 rows × 7 columns

In [42]:
```
df_fin['Rank'] = df_fin.signup_timestamp.rank(method='dense').astype(int
)
```

In [43]:
```
df_fin.drop('signup_timestamp', axis = 1).head(5)
```

Out[43]:

	city_name	signup_os	signup_channel	vehicle_make	vehicle_year	dep_var	Rank
0	Strark	other	R2D	Volkswagen	2012	0	6
1	Berton	ios web	Dost	Toyota	2003	1	3
2	Berton	other	R2D	Hyundai	2015	0	10
3	Berton	ios web	Referral	Honda	2017	1	27
4	Wrouver	other	Referral	Honda	1999	0	17

In [44]:
```
#Changing city name, signup_os, signup_channel, vehicle_make to numerica
l using one hot encoding:
cols = pd.get_dummies(data=df_fin, columns=['city_name', 'signup_os', 's
ignup_channel', 'vehicle_make'])
df_fin[cols.columns] = cols
```

In [45]:
```
#Predictive Modelling:
df_fin.head(10)
```

Out[45]:

	city_name	signup_os	signup_channel	signup_timestamp	vehicle_make	vehicle_year
0	Strark	other	R2D	2017-07-06	Volkswagen	2012
1	Berton	ios web	Dost	2017-07-03	Toyota	2003
2	Berton	other	R2D	2017-07-10	Hyundai	2015
3	Berton	ios web	Referral	2017-07-27	Honda	2017
4	Wrouver	other	Referral	2017-07-17	Honda	1999
5	Berton	ios web	Referral	2017-07-11	other	2010
6	Berton	ios web	Referral	2017-07-13	other	2002
7	Berton	other	R2D	2017-07-28	Mazda	2003
8	Wrouver	android web	Referral	2017-07-08	other	2006
9	Strark	android web	other	2017-07-22	Toyota	2017

10 rows × 37 columns

In [46]:
```
df_fin.columns
```

Out[46]:
```
Index(['city_name', 'signup_os', 'signup_channel', 'signup_timestamp',
       'vehicle_make', 'vehicle_year', 'dep_var', 'Rank', 'city_name_Be
rton',
       'city_name_Strark', 'city_name_Wrouver', 'signup_os_android we
b',
       'signup_os_ios web', 'signup_os_mac', 'signup_os_other',
       'signup_os_windows', 'signup_channel_Dost', 'signup_channel_Orga
nic',
       'signup_channel_Paid', 'signup_channel_R2D', 'signup_channel_Ref
erral',
       'signup_channel_other', 'vehicle_make_BMW', 'vehicle_make_Chevro
let',
       'vehicle_make_Dodge', 'vehicle_make_Ford', 'vehicle_make_Honda',
       'vehicle_make_Hyundai', 'vehicle_make_Jeep', 'vehicle_make_Kia',
       'vehicle_make_Lexus', 'vehicle_make_Mazda',
       'vehicle_make_Mercedes-Benz', 'vehicle_make_Nissan',
       'vehicle_make_Toyota', 'vehicle_make_Volkswagen', 'vehicle_make_
other'],
      dtype='object')
```

In [47]:
```
#Subsetting the dataset for predictive modelling:
X = df_fin.loc[:, ['vehicle_year', 'Rank', 'city_name_Berton',
       'city_name_Strark', 'city_name_Wrouver', 'signup_os_android web',
       'signup_os_ios web', 'signup_os_mac', 'signup_os_other',
       'signup_os_windows', 'signup_channel_Dost', 'signup_channel_Organ
ic',
       'signup_channel_Paid', 'signup_channel_R2D', 'signup_channel_Refe
rral',
       'signup_channel_other', 'vehicle_make_BMW', 'vehicle_make_Chevrol
et',
       'vehicle_make_Dodge', 'vehicle_make_Ford', 'vehicle_make_Honda',
       'vehicle_make_Hyundai', 'vehicle_make_Jeep', 'vehicle_make_Kia',
       'vehicle_make_Lexus', 'vehicle_make_Mazda',
       'vehicle_make_Mercedes-Benz', 'vehicle_make_Nissan',
       'vehicle_make_Toyota', 'vehicle_make_Volkswagen', 'vehicle_make_o
ther']]
X = pd.DataFrame(X)

y = df_fin.loc[:, ['dep_var']]
```

In [48]:
```
# Scaling the 'vehicle_year', 'Rank' columns before predictive modellin
g:
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X[['vehicle_year', 'Rank']] = X[['vehicle_year', 'Rank']] = scaler.fit_t
ransform(X[['vehicle_year', 'Rank']])
X.head(4)
```

Out[48]:

	vehicle_year	Rank	city_name_Berton	city_name_Strark	city_name_Wrouver	signu
0	0.76	0.166667	0	1	0	0
1	0.40	0.066667	1	0	0	0
2	0.88	0.300000	1	0	0	0
3	0.96	0.866667	1	0	0	0

4 rows × 31 columns

In [49]:
```
from sklearn.model_selection import train_test_split
# implementing train-test-split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25
, random_state=66)
```

----------Logistic Model----------

In [66]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\utils\validation.py:578: DataConversionWarning: A column
-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[66]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0, warm_start=False)
```

In [67]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.f
ormat(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.57
```

In [68]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[ 490  743]
 [ 449 1117]]
```

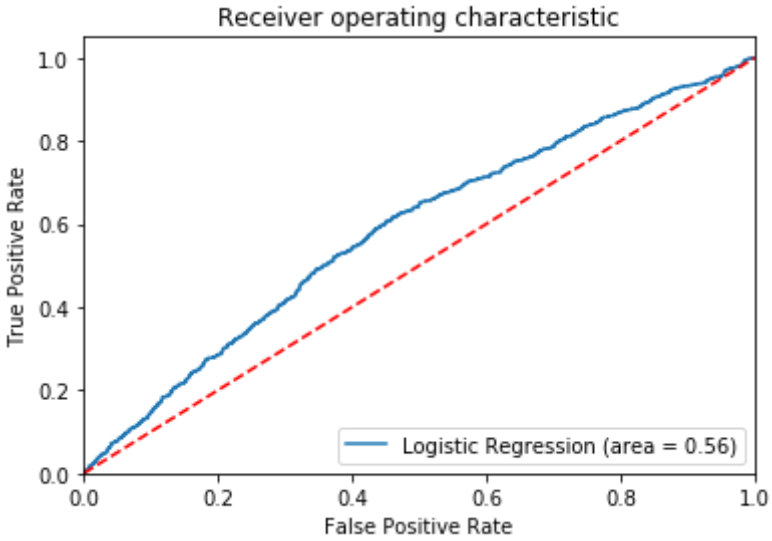
In [53]:
```
# We can see that our model was able to predict 490+1117 correctly and 4
49+743 incorrectlyThe precision is the ratio tp / (tp + fp) where tp is
 the number of true positives and fp the number of false positives. The
 precision is intuitively the ability of the classifier to not label a s
ample as positive if it is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true po
sitives and fn the number of false negatives. The recall is intuitively
the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the p
recision and recall, where an F-beta score reaches its best value at 1 a
nd worst score at 0.

The F-beta score weights the recall more than the precision by a factor
of beta. beta = 1.0 means recall and precision are equally important.
```

In [54]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.52      0.40      0.45      1233
          1       0.60      0.71      0.65      1566

avg / total       0.57      0.57      0.56      2799
```

Of the entire test set, 57% of the promoted term deposit were the term deposit that the customers liked. Of the entire test set, 57% of the customer's preferred term deposits that were promoted

In [55]:
```
#ROC Curve

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,
1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_ro
c_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



The red dotted line represents results of a random classifier; a good classifier stays as far away as possible from the red line. Here we can see that blue line is not very far away from the rod dotted line and thus there is a scope to improve the model

In [ ]:
```
------------------------------Random Forest ------------------------------
--------
```

In [78]:
```
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train.values.ravel())
# predictions
rfc_predict = rfc.predict(X_test)
```

In [79]:
```
#Evaluating the performance:
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix

#Running 10 kross cross validation:
rfc_cv_score = cross_val_score(rfc, X, y, cv=10, scoring='roc_auc')
```

```
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [80]:
```
print("=== Confusion Matrix ===")
print(confusion_matrix(y_test, rfc_predict))
print('\n')
print("=== Classification Report ===")
print(classification_report(y_test, rfc_predict))
print('\n')
print("=== All AUC Scores ===")
print(rfc_cv_score)
print('\n')
print("=== Mean AUC Score ===")
print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
```

```
=== Confusion Matrix ===
[[667 566]
 [700 866]]


=== Classification Report ===
             precision    recall  f1-score   support

          0       0.49      0.54      0.51      1233
          1       0.60      0.55      0.58      1566

avg / total       0.55      0.55      0.55      2799



=== All AUC Scores ===
[ 0.50895436  0.52632375  0.5217371   0.50846494  0.52187877  0.5450219
4
  0.5207801   0.5248645   0.55384566  0.52054088]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.525241201079
```

We'll evaluate our model's score based on the roc_auc score, which is 52.3% The accuracy achievde is not very good so we can do hyperparameter tunning to see if we can improve the results We can see that our predictor predicted 645+902 correctly and 664+588 incorrectly

In [168]:
```
from sklearn.model_selection import RandomizedSearchCV
# number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, nu
m = 10)]
# number of features at every split
max_features = ['auto', 'sqrt']

# max depth
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)
# create random grid
random_grid = {
 'n_estimators': n_estimators,
 'max_features': max_features,
 'max_depth': max_depth
 }
# Random search of parameters
rfc_random = RandomizedSearchCV(estimator = rfc, param_distributions = r
andom_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -
1)
# Fit the model
rfc_random.fit(X_train, y_train)
# print results
print(rfc_random.best_params_)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:  9.2min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 18.3min finished
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_search.py:739: 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().
  self.best_estimator_.fit(X, y, **fit_params)

{'max_depth': 380, 'n_estimators': 600, 'max_features': 'auto'}
```

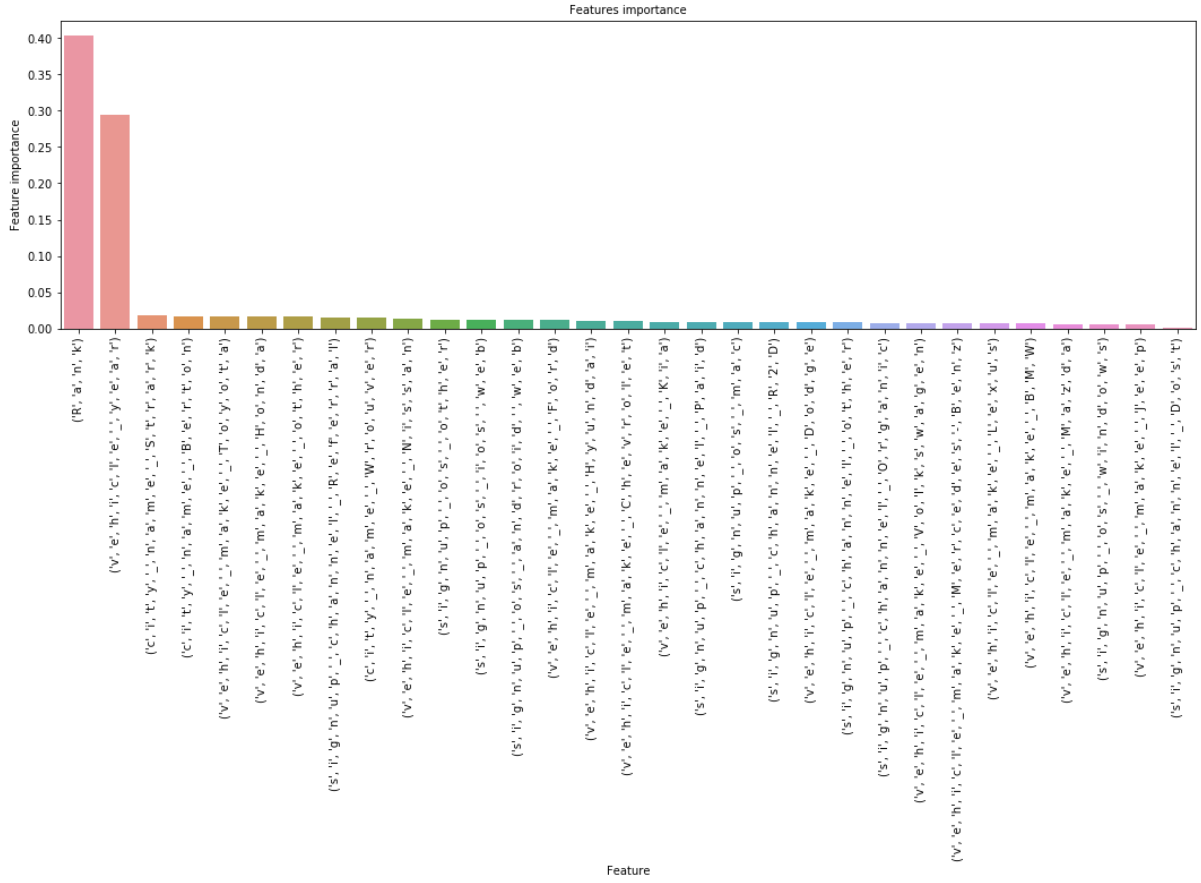
In [81]:
```
rfc = RandomForestClassifier(n_estimators=600, max_depth=380, max_featur
es='auto')
rfc.fit(X_train,y_train)
rfc_predict = rfc.predict(X_test)
rfc_cv_score = cross_val_score(rfc, X, y, cv=10, scoring='roc_auc')
print("=== Confusion Matrix ===")
print(confusion_matrix(y_test, rfc_predict))
print('\n')
print("=== Classification Report ===")
print(classification_report(y_test, rfc_predict))
print('\n')
print("=== All AUC Scores ===")
print(rfc_cv_score)
print('\n')
print("=== Mean AUC Score ===")
print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
```

```
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\ipykernel\__main__.py:2: 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().
  from ipykernel import kernelapp as app
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\abhip\AppData\Local\Enthought\Canopy\edm\envs\User\lib\site-pa
ckages\sklearn\model_selection\_validation.py:458: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
=== Confusion Matrix ===
[[606 627]
 [619 947]]


=== Classification Report ===
             precision    recall  f1-score   support

          0       0.49      0.49      0.49      1233
          1       0.60      0.60      0.60      1566

avg / total       0.55      0.55      0.55      2799



=== All AUC Scores ===
[ 0.5052113   0.53171859  0.52480077  0.51593979  0.5323207   0.5514389
  0.52679217  0.52045909  0.5545764   0.52694567]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.529020336838
```

We can see that there is not much of the improvement in the model after hyperparameter tunning as well.

In [75]:
```
tmp = pd.DataFrame({'Feature': X, 'Feature importance': rfc.feature_impo
rtances_})
tmp = tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize = (18,5))
plt.title('Features importance',fontsize=10)
s = sns.barplot(x='Feature',y='Feature importance',data= tmp)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```



- Uber should focus on drivers with newer cars. It may help giving bonuses or other forms of incentives to the this type of drivers
- Uber can reallocate the resources and money to award signup referrals. This will not only grow the Uber driver network, also those who signup via referral are more likely to take a first trip
- Uber should keep reminding the signed-up drivers whether they wish to take a first trip. It can be helpful to follow up with the potential drivers, providing them useful information to get them on the road
- We can also see that R2D & Paid channel have potential for growth so we can probably think of shifting our focus in terms of budgeting and other efforts from dost to these 2 channels more

- Wrouver city has the lowest share in the overall drivers turnout to drive with Uber so we can see if we can shift focus on a similr city or if we need to workaround to make things better at Wrouver than we can probably make pricing/policy changes
- Strark has the best turnout from all the channels with best from Referral channel and we can focus more on this city as potential is high in this city
- We can see that the other category is the highest when categorized under signup_os drivers fall into. This shows that the team responsible for taking a note of signup_os or people filling automatic forms are excluding this particular information. We should make it mandatory to get this field filled
- Windows and Mac have the lowest share showing that most of the drivers coming to register do it more on their mobiles given the other category fall in the same weightage as right now if we exclude otehr category. We can probbaly make a budget cut on these 2 channels and focus more on mobile platforms
- Ios appears to get the most customers and thus we can focus more on this channel in future if we need instant growth
- We can see a clear trend that Honda & Toyota brands are more likely to make it through and driver for Uber so we can probbaly focus more on
- Sedan type and we can also check the pricing model for Sedan and try to implement other types models also in similar way
- Nissan also looks like a potential brand on which we can put more focus in future
- AUC score, Roc curve, F1 score are some of the relevant factors to check how effective our model is. As we can see that our model was not able to predict with a lot of accuracy. However, it does not suggest that the model build was a bad model but it can also be because of the lack of features and amount of datset required to implement a better model. We would probbaly need more driver related stats like what is his credit score, what was his occupation earlier etc to made a better predictive model
- I was a little concerned about the missing dates from the respective date columns and also about the fact that most of the signup and bgc dates were equivalent to each other.
- Vehicle year and dates when the vehicle was registered for signup or signup time along with signup channel and city were the most important features in the model
- Improvements:

1. More Complex Models - The models I evaluated were fairly simple models. Maybe a more complex model would have performed better. For example, an ensemble method such as Gradient Boosting Machines wand a Neural Network would have predicted in a better way
2. We could have done a chi square test and tested variables more to get a more significance of the variables