Bike Share

June 6, 2019

1 Boston Bike Share Data

- **See the main project notebook for instructions to be sure you satisfy the rubric!**
- See project 03 for information on the dataset.
- A few example prediction questions to pursue are listed below. However, don't limit yourself to them!
  - Predict the trip duration.
  - Guess the age or gender of a user. (e.g. for null-imputation with a model).
  - Guess whether a given ride will be point-to-point or round-trip (be careful not to use "future" information!).

Be careful to justify what information you would know at the "time of prediction" and train your model using only those features.

we want to predict the duration of trip form the dataset.

1.1 ### Results: Baseline Model

In order to predict the tripduration, we want to conclude from three evaluation metrics, gender, birth year and usertype. Besides, we need to transform these features first

```
For gender: because male are more willing to take blue bikes (accoring to proj3), male may pre

For usertype: there are two kinds of usertypes, subscriber and customers. From our perspective

Therefore, because both usertypes and gender are categorical features, we use one-hot-encoder t


For age: we think because younger people may be more energetic, they probably prefer to take bl

Therefore, because birth year is quantitative feature, we apply non-linear transformation to it
result:
score: 0.002306310040886106
RMSE: 31900.791241128314
```

1.2 ### Results: Final Model

In the final model, we have added the three new featuers: 'age', 'work', 'distance' the age feature stands for the age of each user, which derived from the birthyear clumn; the work feature determines if the trip is for work purpose or not; the distance featuer is derived from the longitude and latitude from the start point and the end ponit. both of these 3 features are relevant to determine the duration of trip. For the model, we choose these three features, and the gender, age, and we choose the linear regression as the model we use.

result: score:0.002839332531130556 rmse31892.26852966829

1.3 ### Results: Fairness Evaluation

1.3.1 Evaluate model for fairness

NUll hypo: it's fair to evaluate on gender ALTER hypo: it's not fair to evaluate on gender

For evaluating the fairness on 'gender', we use permutation test.

However, the p-val is larger tha significant level 0.05, which means that we have to reject the null hypo that this model is fair on gender.

---

2 Your Code Starts Here

```
In [1]: %matplotlib inline
        import os

        import pandas as pd
        import numpy as np

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

```
In [37]: from sklearn.linear_model import LinearRegression
         from sklearn.preprocessing import FunctionTransformer
         from sklearn.preprocessing import OneHotEncoder
         from sklearn.pipeline import Pipeline
         from sklearn.compose import ColumnTransformer

         from sklearn.tree import DecisionTreeRegressor
         from sklearn.model_selection import train_test_split
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeClassifier

         import math
         from sklearn.datasets import load_breast_cancer
         from sklearn.linear_model import LogisticRegression
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

2.1 ### Baseline

State the prediction problem you are attempting, if it's a classification or regression problem, explain your choice of target variable and evaluation metric (objective).

Attempt 1: Considering gender, usertype and the purposes of commuting by blue bikes (for working or not working), age and tripduration, we use various models to predict the distance.

Nomial:gender, usertype, work.

Quantitative: age, tripduration

```
In [98]: ct = ColumnTransformer([('ohe', OneHotEncoder(), ['gender', 'usertype']),\
                                ('log', FunctionTransformer(np.log,validate=False), ['birth yea
                               remainder='passthrough')

         pl=Pipeline([('column', ct), ('lin-reg', LinearRegression())])
```

```
In [105]: pl.fit(df[['gender', 'usertype', 'birth year',
                    'start station latitude', 'start station longitude',
                    'end station latitude', 'end station longitude']], df.tripduration)
          pl.score(df[['gender', 'usertype', 'birth year',
                      'start station latitude', 'start station longitude',
                    'end station latitude', 'end station longitude']], df.tripduration)
```

```
Out[105]: 0.002306310040886106
```

```
In [106]: rmse(pl.predict(df[['gender', 'usertype', 'birth year',
                              'start station latitude', 'start station longitude',
                    'end station latitude', 'end station longitude']]), df.tripduration)
```

```
Out[106]: 31900.791241128314
```

```
In [ ]:
```

2.2 ### Final Model

2.3 #### 1.clean up data

```
In [108]: #combine small csv files into one large dataframe
          df=pd.DataFrame()
          for i in range(5,16):
              if i < 10:
                  file = '180%s'%i
              elif i >= 13:
                  file = '190{}'.format(i-12)
              else:
                  file = '18{}'.format(i)
              fp = os.path.join('data','{}.csv'.format(file))
              df = pd.concat([df,pd.read_csv(fp)])
```

In [4]:
```
print(df.shape)
df.head(5)
```

```
(1755585, 15)
```

Out[4]:

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year	gender
0	1177	2018-05-01 00:01:32.4590	2018-05-01 00:21:10.0260	184	Sidney Research Campus/ Erie Street at Waverly	42.357753	-71.103934	189	Kendall T	42.362428	-71.084955	790	Subscriber	1994	1
1	733	2018-05-01 00:05:19.4970	2018-05-01 00:17:32.7190	67	MIT at Mass Ave / Amherst St	42.358100	-71.093198	41	Packard's Corner - Commonwealth Ave at Brighto...	42.352261	-71.123831	1238	Subscriber	1993	2
2	437	2018-05-01 00:05:37.7590	2018-05-01 00:12:54.8300	54	Tremont St at West St	42.354979	-71.063348	6	Cambridge St at Joy St	42.361291	-71.065262	218	Subscriber	1993	1
3	730	2018-05-01 00:05:39.6780	2018-05-01 00:17:50.5880	54	Tremont St at West St	42.354979	-71.063348	46	Christian Science Plaza - Massachusetts Ave at...	42.343666	-71.085824	1885	Subscriber	1992	1
4	411	2018-05-01 00:06:10.1590	2018-05-01 00:13:02.0490	54	Tremont St at West St	42.354979	-71.063348	6	Cambridge St at Joy St	42.361291	-71.065262	602	Customer	1969	0

In [5]:
```
#data_types
quan = ['tripduaration', 'startime', 'stoptime', 'start station latitude',
       'start station longitude', 'end station latitude', 'end station longitude',
       'birth year']
ordinal = ['usertype', 'gender']
nominal = ['start station id', 'start station name', 'end station id',
          'end station name','bikeid']
```

In [ ]:

2.4 #### 2.add new features into the original dataframe

first create a helper method to convert the birth year to age according to the year recored, then add the age feature to the dataset

In [109]:
```
# find the age from calculating the birth year and starttime columns
def find_year(x):
    return (x['starttime'].str)[:4][:,None]
ct=ColumnTransformer([('year', FunctionTransformer(find_year,validate=False),['start
ct.fit(df)

# add the age column to the dataset
df['age']=ct.transform(df)
df['age']=(df.age).astype(int).subtract(df['birth year'])
print(df.shape)
df.head(2)
```

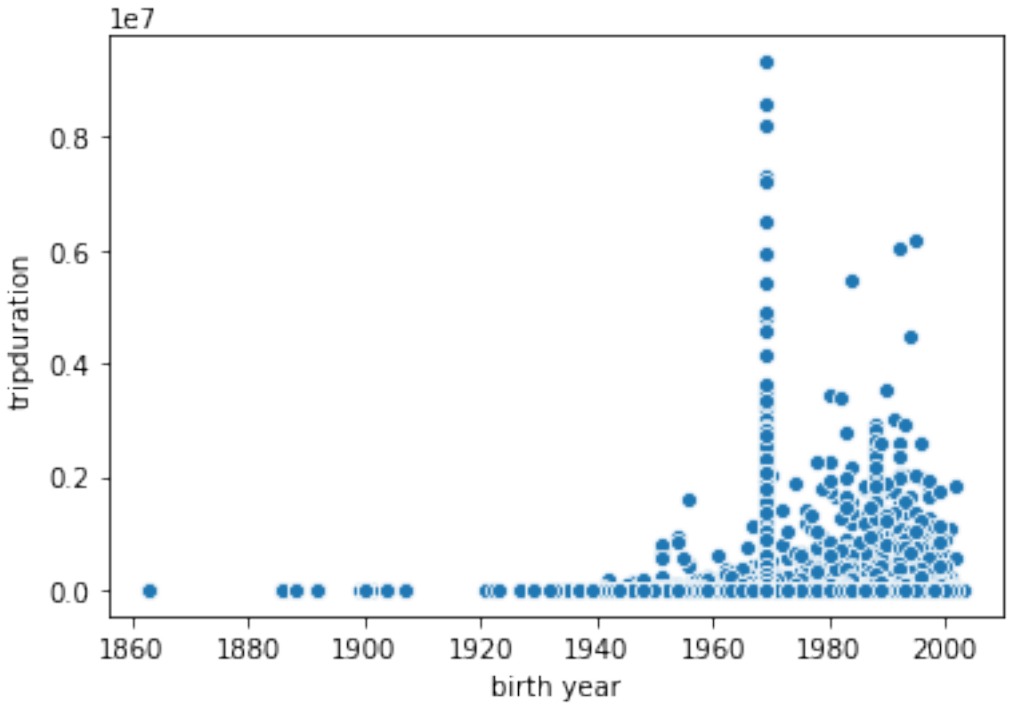
```
(1755585, 16)
```

Out[109]:

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year	gender	age
0	1177	2018-05-01 00:01:32.4590	2018-05-01 00:21:10.0260	184	Sidney Research Campus/ Erie Street at Waverly	42.357753	-71.103934	189	Kendall T	42.362428	-71.084955	790	Subscriber	1994	1	24
1	733	2018-05-01 00:05:19.4970	2018-05-01 00:17:32.7190	67	MIT at Mass Ave / Amherst St	42.358100	-71.093198	41	Packard's Corner - Commonwealth Ave at Brighto...	42.352261	-71.123831	1238	Subscriber	1993	2	25

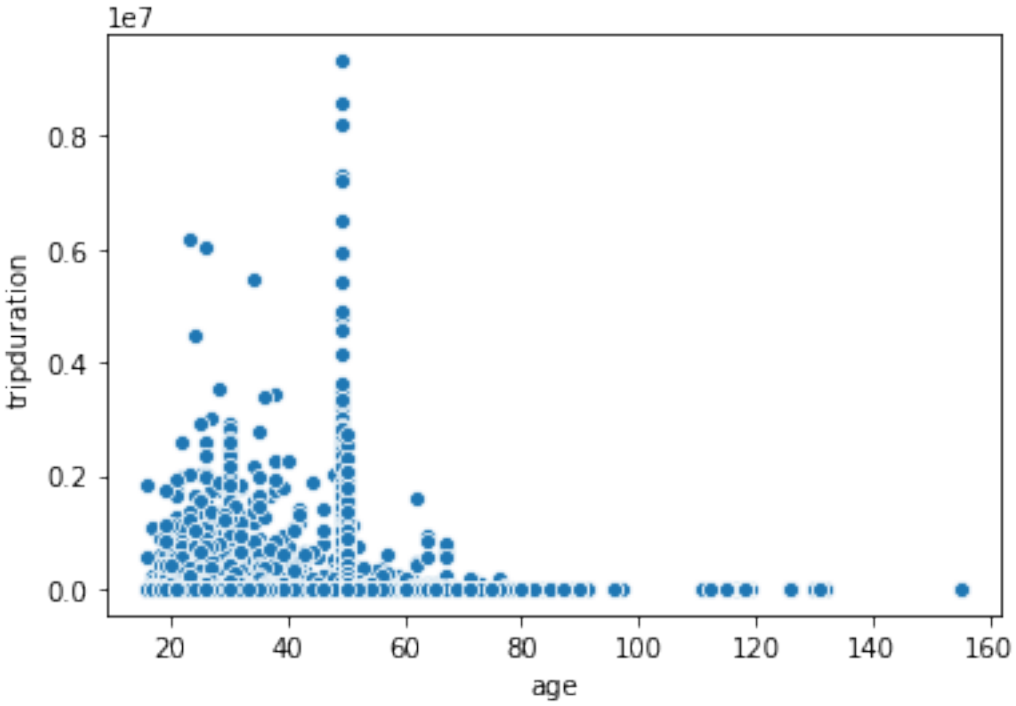
In [7]:
```
sns.scatterplot(df['birth year'], df['tripduration'])
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x270e51786d8>
```



In [9]:
```
sns.scatterplot(df['age'], df['tripduration'])
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd0773dc50>



In [ ]:

then engineer the new feature 'distance' which derived from the columns 'start station latitude', 'start station longitude' and 'end station latitude', 'end station longitude'

In [110]:
```
# define a helper function to find the distance from two points
# using latitude and longitude

def distance(lat1, lon1, lat2, lon2):
    radius = 6371 #in km unit
    dlat = np.radians(lat2 - lat1)
    dlon = np.radians(lon2 - lon1)
    a = (np.sin(dlat / 2) * np.sin(dlat / 2) +
         np.cos(np.radians(lat1)) * np.cos(np.radians(lat2)) *
         np.sin(dlon / 2) * np.sin(dlon / 2))
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    d = radius * c

    return d

def find_distance(x):
    return distance(x['start station latitude'], x['start station longitude'],
                    x['end station latitude'], x['end station longitude'])[:,None]
```

In [111]:
```
# apply this method to the four corresponding columns to
# get the new feature
ct = ColumnTransformer([('distance',
                         FunctionTransformer(find_distance,validate=False),
                        ['start station latitude', 'start station longitude',
                        'end station latitude', 'end station longitude'])])
ct.fit(df)
df['distance'] = ct.transform(df)
print(df.shape)
df.head(2)
```

```
(1755585, 17)
```

Out[111]:

	tripduration	starttime	stoptime	start station id	start station name
0	1177	2018-05-01 00:01:32.4590	2018-05-01 00:21:10.0260	184	Sidney Research Campus/ Erie Street at Waverly
1	733	2018-05-01 00:05:19.4970	2018-05-01 00:17:32.7190	67	MIT at Mass Ave / Amherst St

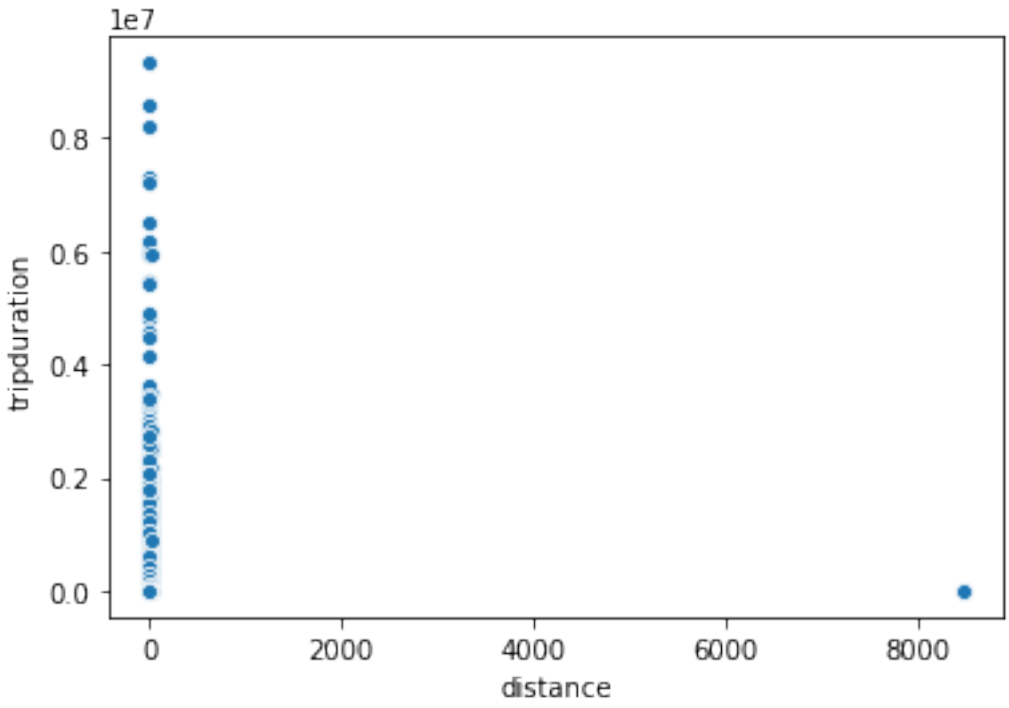
```
   start station latitude  start station longitude  end station id  \
0               42.357753               -71.103934             189
1               42.358100               -71.093198              41

                                      end station name  end station latitude  \
0                                           Kendall T             42.362428
1  Packard's Corner - Commonwealth Ave at Brighto...             42.352261

   end station longitude  bikeid    usertype  birth year  gender  age  \
0             -71.084955     790  Subscriber        1994       1   24
1             -71.123831    1238  Subscriber        1993       2   25

   distance
0  1.643782
1  2.599535
```

In [10]:
```
## to see the new feature visually
df.sort_values('distance',ascending=False).head(4)
```

Out[10]:
```
        tripduration                starttime                 stoptime  \
73497             66  2018-11-15 15:59:08.1960  2018-11-15 16:00:14.7030
164924           124  2018-10-25 11:01:43.6110  2018-10-25 11:03:47.7280
50153             66  2019-01-23 16:44:09.1690  2019-01-23 16:45:15.2490
34024          33092  2018-11-07 12:31:42.4910  2018-11-07 21:43:14.6080

        start station id start station name  start station latitude  \
73497                229   8D QC Station 01               42.345033
164924               229   8D QC Station 01               42.345033
50153                229   8D QC Station 01               42.345033
34024                308       BCBS Hingham               42.167226

        start station longitude  end station id       end station name  \
73497                -71.096649             230       8D QC Station 02
164924               -71.096649             230       8D QC Station 02
50153                -71.096649             230       8D QC Station 02
34024                -70.905558               1  18 Dorrance Warehouse

        end station latitude  end station longitude  bikeid    usertype  \
73497               0.000000               0.000000    1583    Customer
164924              0.000000               0.000000    1583    Customer
50153               0.000000               0.000000    1583    Customer
34024              42.387151             -71.075978    4254  Subscriber

        birth year  gender  age     distance
73497         1969       0   49  8467.047261
164924        1969       0   49  8467.047261
50153         1969       0   50  8467.047261
34024         1988       2   30    28.188901
```

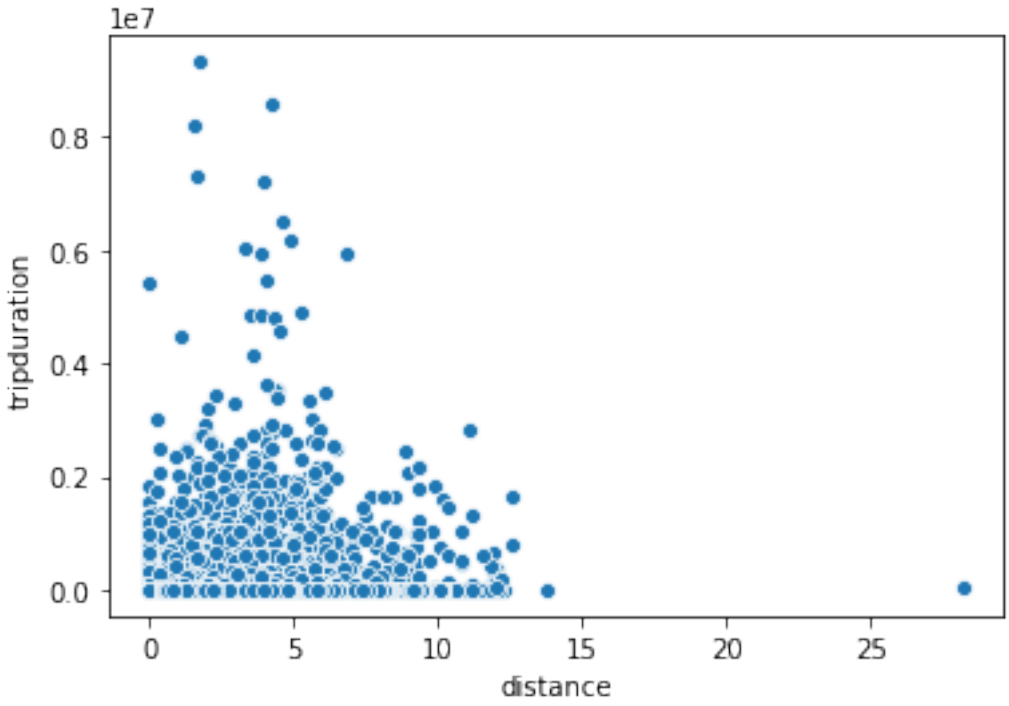
```
In [11]: sns.scatterplot(df['distance'], df['tripduration'])
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x270854c5390>
```




```
In [18]: sns.scatterplot(df['distance'].drop([73497,164924,50153],axis=0), df['tripduration'].
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd07865898>
```



create a new feature 'work' which indicates if this trip occurred during weekdays or not, and if during the worktime or not

```
In [112]: # to determine if the trip is for work purpose
          weekday_bool=pd.to_datetime(df['starttime']).dt.dayofweek<=5
          stoptime_bool=(pd.to_datetime(df['stoptime']).dt.strftime('%H:%M')<'09:30')&\
                        (pd.to_datetime(df['stoptime']).dt.strftime('%H:%M')>'07:00')
          trip_bool=df['tripduration']<3600
          work_bool=weekday_bool&stoptime_bool&trip_bool

          # add the bool array to the dataset
          df['work'] = work_bool
```

```
In [13]: print(df.shape)
         df.head(2)
```

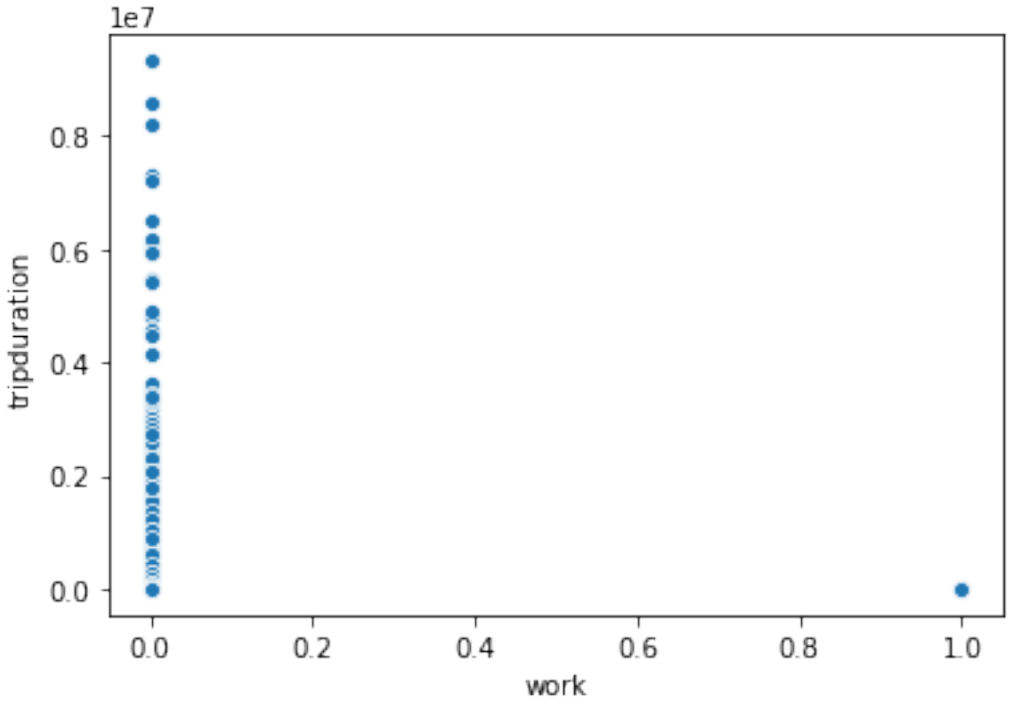
```
(1755585, 18)
```

```
Out[13]:    tripduration                starttime                 stoptime  \
         0          1177  2018-05-01 00:01:32.4590  2018-05-01 00:21:10.0260
         1           733  2018-05-01 00:05:19.4970  2018-05-01 00:17:32.7190

            start station id                            start station name  \
         0               184  Sidney Research Campus/ Erie Street at Waverly
```

```
1                67                    MIT at Mass Ave / Amherst St

   start station latitude  start station longitude  end station id  \
0               42.357753               -71.103934             189
1               42.358100               -71.093198              41

                                    end station name  end station latitude  \
0                                        Kendall T             42.362428
1  Packard's Corner - Commonwealth Ave at Brighto...             42.352261

   end station longitude  bikeid    usertype  birth year  gender  age  \
0             -71.084955     790  Subscriber        1994       1   24
1             -71.123831    1238  Subscriber        1993       2   25

   distance   work
0  1.643782  False
1  2.599535  False
```

In [21]:
```
sns.scatterplot(df['work'], df['tripduration'])
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd078b03c8>
```



In [ ]:

2.5 #### 3.putting these new features together to predict

```
In [113]: def rmse(predictions, targets):
              return np.sqrt(((predictions - targets) ** 2).mean())

In [131]: ct = ColumnTransformer([('ohe', OneHotEncoder(), ['gender', 'usertype', 'work'])],
                                 remainder='passthrough')
          pl1 = Pipeline([('ct', ct),
                          ('lin-reg', LinearRegression())])

In [142]: features=df.drop(['starttime', 'stoptime', 'start station name',
                            'end station name','tripduration','start station id',
                             'end station id', 'bikeid'], axis=1)
          #features=df[['gender', 'usertype', 'work', 'distance', 'age']]
          pl1.fit(features,df.tripduration)
          pred=pl1.predict(features)
          print(pl1.score(features, df.tripduration))
          rmse(pred,df.tripduration)
```

```
0.002839332531130556
```

```
Out[142]: 31892.26852966829
```

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

2.6 ### Inference Evaluation

```
In [ ]: df.head()
        df['prediction']=new_preds

In [ ]: gender_bool=(df['gender']==1)^(df['gender']==2)
        combined = df.loc[gender_bool]

In [ ]: L = []
        for i in range(100):
            lst=np.random.choice(1533353,100 )
            df=combined.iloc[lst]
            s=(
                df[['gender', 'prediction']]
                .assign(gender=combined.gender.sample(frac=1, replace=False).reset_index(drop=
                .groupby('gender')
```

```
                .prediction
                .apply(np.mean)
                .diff()
                .iloc[-1]

            )
            L.append(s)
```

In [ ]:
```
Lst = [abs(x) for x in L]
Lst
```

In [ ]:
```
observed=combined.groupby('gender')['prediction'].apply(np.mean).diff().iloc[-1]
```

In [ ]:
```
p_val=np.count_nonzero(Lst >= observed) / 100
p_val
```