COSE471 proj2

GAOZHONGSONG

TOTAL POINTS

22 / 22

QUESTION 1

1 Q1a **1 / 1**

✓ **+ 1 pts** **Correct**

+ 0 pts Wrong

+ 0.75 pts partially correct 75%

+ 0.67 pts partially correct 67%

+ 0.5 pts partially correct 50%

+ 0.25 pts partially correct 25%

QUESTION 2

2 Q1b **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 0 pts partially correct

QUESTION 3

3 Q2a **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1 pts partially correct

QUESTION 4

4 Q2b **1 / 1**

✓ **+ 1 pts** **Correct**

+ 0 pts Wrong

+ 0.5 pts partially correct

QUESTION 5

5 Q2c **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1 pts partially correct

QUESTION 6

6 Q2d **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1.5 pts partially correct 75%

+ 1 pts partially correct 50%

+ 1.7 pts partially correct 86%

+ 1.6 pts partially correct 80%

QUESTION 7

7 Q3a **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1 pts partially correct 50%

QUESTION 8

8 Q3b **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1 pts partially correct

QUESTION 9

9 Q3c **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1.5 pts partially correct 75%

+ 1.3 pts partially correct 67%

+ 1 pts partially correct 50%

QUESTION 10

10 Q3d **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

+ 1 pts partially correct 50%

+ 1.3 pts partially correct 67%

QUESTION 11

11 Q3e **2 / 2**

✓ **+ 2 pts** **Correct**

+ **0 pts** Wrong

QUESTION 12

12 Q3f **2 / 2**

✓ **+ 2 pts** **Correct**

+ **0 pts** Wrong

+ **1.5 pts** partially correct 75%

+ **1 pts** partially correct 50%

gradescope

Part 1: Exploratory Data Analysis

In this part, you'll choose which days to include as training data in your regression model.

Your goal is to develop a general model that could potentially be used for future taxi rides. There is no guarantee that future distributions will resemble observed distributions, but some effort to limit training data to typical examples can help ensure that the training data are representative of future observations.

Note that January 2016 had some atypical days.

- New Years Day (January 1) fell on a Friday.
- Martin Luther King Jr. Day was on Monday, January 18.
- A historic blizzard (https://en.wikipedia.org/wiki/January_2016_United_States_blizzard) passed through New York that month.

Using this dataset to train a general regression model for taxi trip times must account for these unusual phenomena, and one way to account for them is to remove atypical days from the training data.

Question 1a

Add a column labeled `date` to `manhattan_taxi` that contains the date (but not the time) of pickup, formatted as a `datetime.date` value (docs (https://docs.python.org/3/library/datetime.html#date-objects)).

The provided tests check that you have extended `manhattan_taxi` *correctly.*

In [22]:
```
# BEGIN YOUR CODE
# ------------------------
manhattan_taxi.loc[:, 'date'] = pd.to_datetime(manhattan_taxi['pickup_datetime']).a
pply(lambda x: x.date())
# ------------------------
# END YOUR CODE
manhattan_taxi.head()
```

Out[22]:

	pickup_datetime	dropoff_datetime	pickup_lon	pickup_lat	dropoff_lon	dropoff_
0	2016-01-30 22:47:32	2016-01-30 23:03:53	-73.988251	40.743542	-74.015251	40.70980
1	2016-01-04 04:30:48	2016-01-04 04:36:08	-73.995888	40.760010	-73.975388	40.78220
2	2016-01-07 21:52:24	2016-01-07 21:57:23	-73.990440	40.730469	-73.985542	40.73851
3	2016-01-08 18:46:10	2016-01-08 18:54:00	-74.004494	40.706989	-74.010155	40.71675
4	2016-01-02 12:39:57	2016-01-02 12:53:29	-73.958214	40.760525	-73.983360	40.76040

In [23]:
```
ok.grade("q1a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 1b

Create a data visualization that allows you to identify which dates were affected by the historic blizzard of January 2016. Make sure that the visualization type is appropriate for the visualized data.

1 Q1a **1 / 1**

✓ **+ 1 pts** **Correct**

+ **0 pts** Wrong

+ **0.75 pts** partially correct 75%

+ **0.67 pts** partially correct 67%

+ **0.5 pts** partially correct 50%

+ **0.25 pts** partially correct 25%

gradescope

In [22]:
```
# BEGIN YOUR CODE
# ------------------------
manhattan_taxi.loc[:, 'date'] = pd.to_datetime(manhattan_taxi['pickup_datetime']).a
pply(lambda x: x.date())
# ------------------------
# END YOUR CODE
manhattan_taxi.head()
```

Out[22]:

	pickup_datetime	dropoff_datetime	pickup_lon	pickup_lat	dropoff_lon	dropoff_
0	2016-01-30 22:47:32	2016-01-30 23:03:53	-73.988251	40.743542	-74.015251	40.70980
1	2016-01-04 04:30:48	2016-01-04 04:36:08	-73.995888	40.760010	-73.975388	40.78220
2	2016-01-07 21:52:24	2016-01-07 21:57:23	-73.990440	40.730469	-73.985542	40.73851
3	2016-01-08 18:46:10	2016-01-08 18:54:00	-74.004494	40.706989	-74.010155	40.71675
4	2016-01-02 12:39:57	2016-01-02 12:53:29	-73.958214	40.760525	-73.983360	40.76040

In [23]:
```
ok.grade("q1a");
```

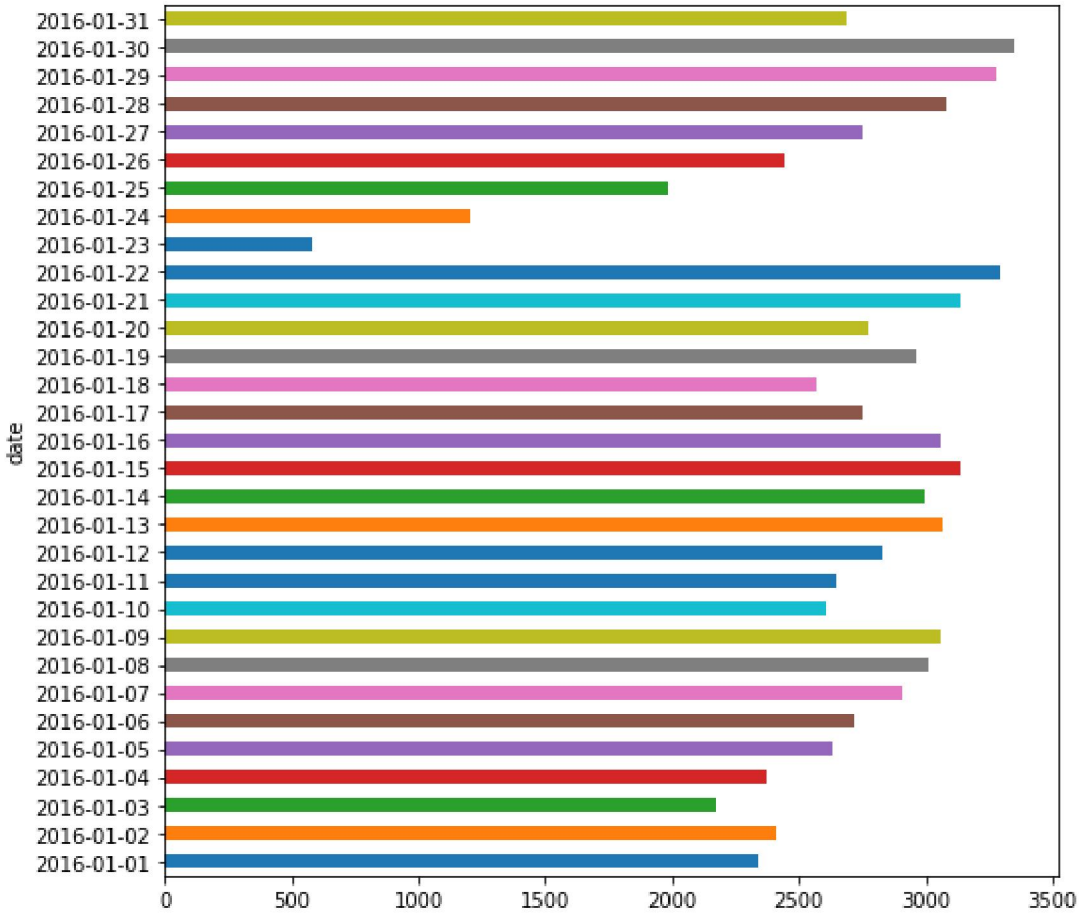
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 1b

Create a data visualization that allows you to identify which dates were affected by the historic blizzard of January 2016. Make sure that the visualization type is appropriate for the visualized data.

In [31]:
```
# BEGIN YOUR CODE
# ------------------------
manhattan_taxi.groupby('date').size().plot(kind='barh', figsize=(8, 8));
# ------------------------
# END YOUR CODE
```



Finally, we have generated a list of dates that should have a fairly typical distribution of taxi rides, which excludes holidays and blizzards. The cell below assigns `final_taxi` to the subset of `manhattan_taxi` that is on these days. (No changes are needed; just run this cell.)

2 Q1b **2 / 2**

✓ **+ 2 pts Correct**

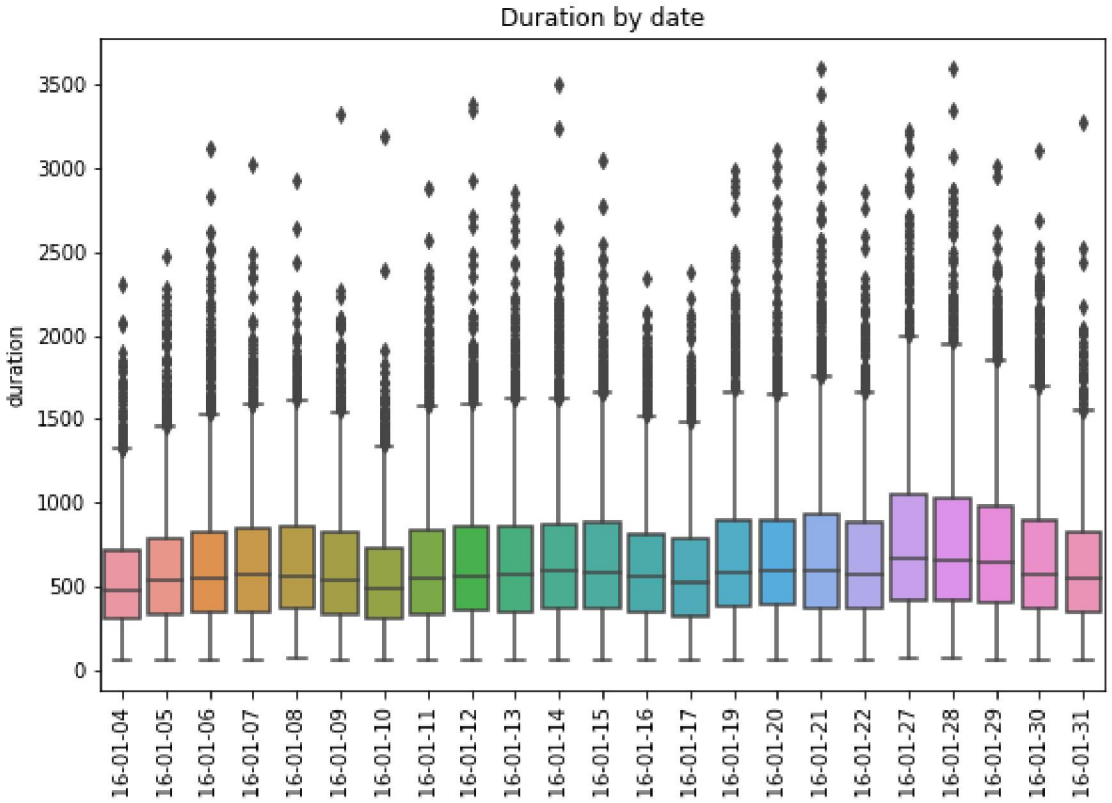
+ 0 pts Wrong

+ 0 pts partially correct

gradescope

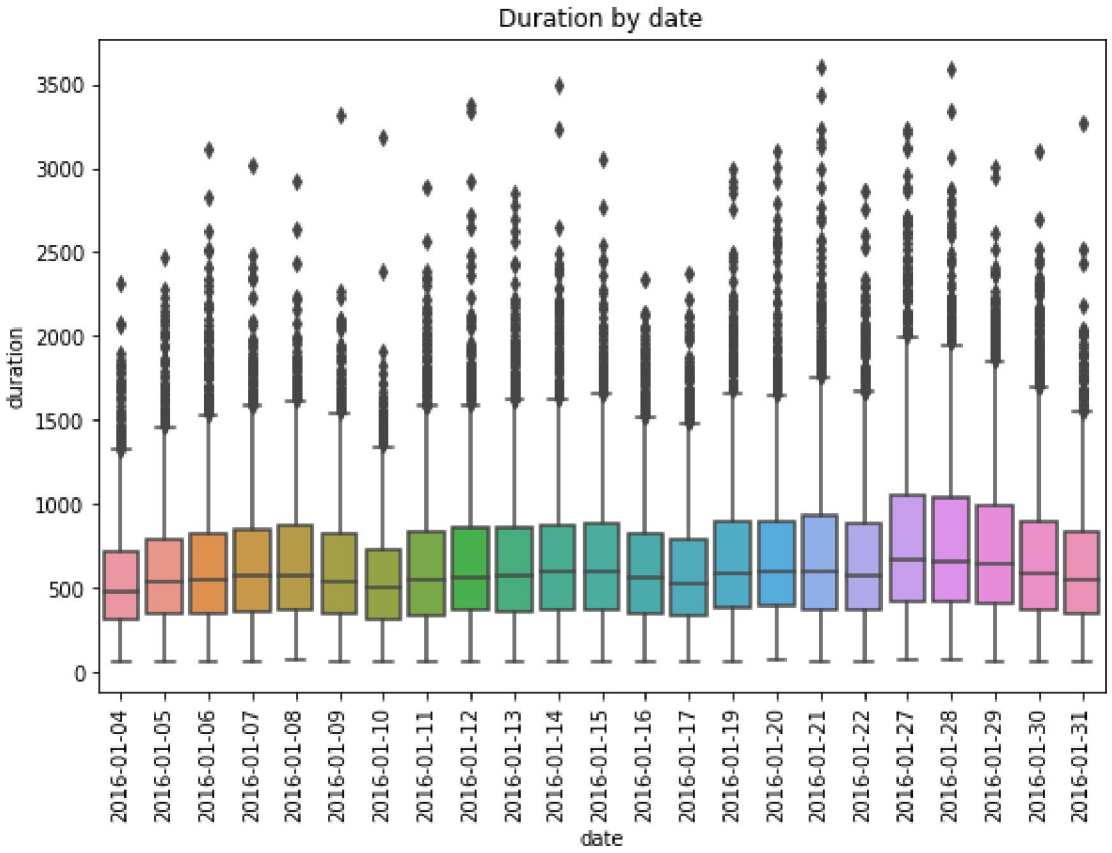
Question 2a

Use `sns.boxplot` to create a box plot that compares the distributions of taxi trip durations for each day **using `train` only**. Individual dates shoud appear on the horizontal axis, and duration values should appear on the vertical axis. Your plot should look like this:



In [34]:
```
plt.figure(figsize=(9, 6))
# BEGIN YOUR CODE
# ------------------------
data = train.sort_values('date')
plt.figure(figsize=(9, 6))
sns.boxplot('date', 'duration', data=data);
plt.xticks(rotation=90);
plt.title('Duration by date');
# ------------------------
# END YOUR CODE
```

```
<matplotlib.figure.Figure at 0x25f8b198d68>
```



Question 2b

In one or two sentences, describe the assocation between the day of the week and the duration of a taxi trip.

Note: The end of Part 2 showed a calendar for these dates and their corresponding days of the week.

3 Q2a **2 / 2**

✓ **+ 2 pts** **Correct**

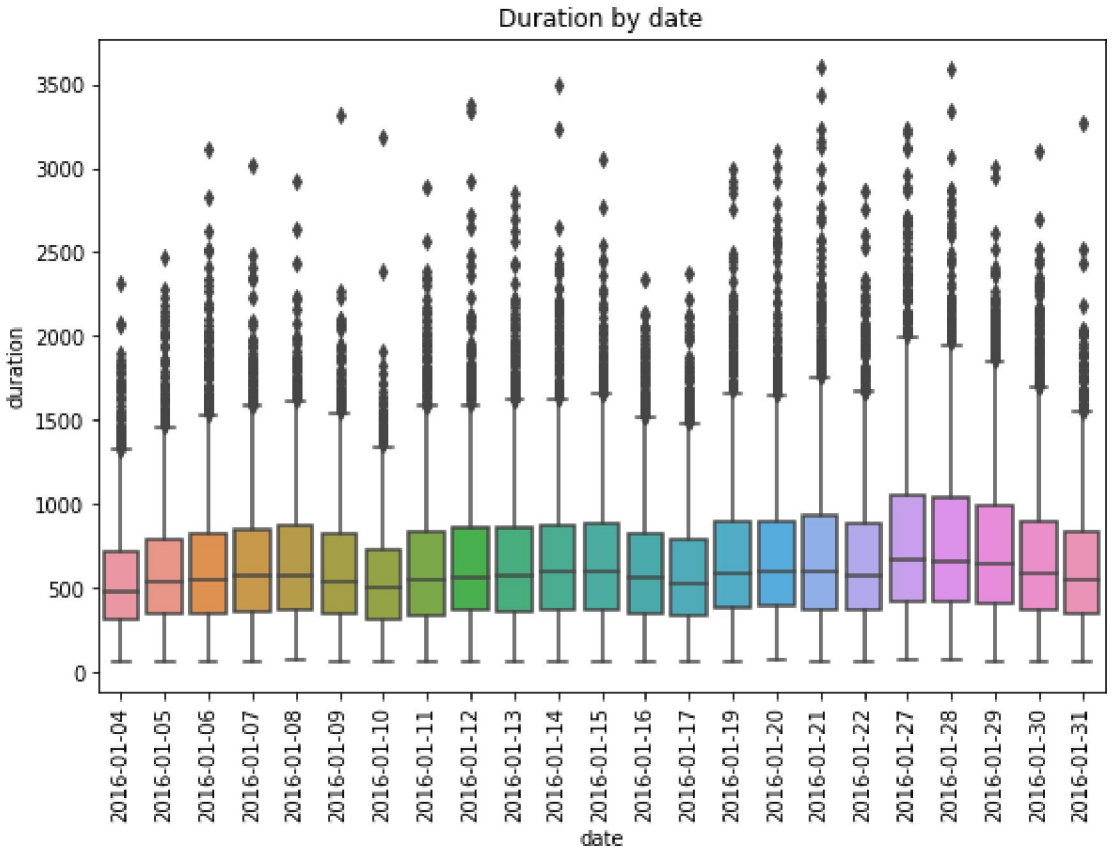
+ 0 pts Wrong

+ 1 pts partially correct

gradescope

In [34]:
```
plt.figure(figsize=(9, 6))
# BEGIN YOUR CODE
# ------------------------
data = train.sort_values('date')
plt.figure(figsize=(9, 6))
sns.boxplot('date', 'duration', data=data);
plt.xticks(rotation=90);
plt.title('Duration by date');
# ------------------------
# END YOUR CODE
```

```
<matplotlib.figure.Figure at 0x25f8b198d68>
```



Question 2b

In one or two sentences, describe the assocation between the day of the week and the duration of a taxi trip.

Note: The end of Part 2 showed a calendar for these dates and their corresponding days of the week.

Answer: The mean of the duration of taxi trips in weekdays is usually higher than the mean of the duration o ftaxi trips in the weekends.

Below, the provided `augment` function adds various columns to a taxi ride dataframe.

- `hour`: The integer hour of the pickup time. E.g., a 3:45pm taxi ride would have `15` as the hour. A 12:20am ride would have `0`.
- `day`: The day of the week with Monday=0, Sunday=6.
- `weekend`: 1 if and only if the `day` is Saturday or Sunday.
- `period`: 1 for early morning (12am-6am), 2 for daytime (6am-6pm), and 3 for night (6pm-12pm).
- `speed`: Average speed in miles per hour.

No changes are required; just run this cell.

In [35]:
```
def speed(t):
    """Return a column of speeds in miles per hour."""
    return t['distance'] / t['duration'] * 60 * 60

def augment(t):
    """Augment a dataframe t with additional columns."""
    u = t.copy()
    pickup_time = pd.to_datetime(t['pickup_datetime'])
    u.loc[:, 'hour'] = pickup_time.dt.hour
    u.loc[:, 'day'] = pickup_time.dt.weekday
    u.loc[:, 'weekend'] = (pickup_time.dt.weekday >= 5).astype(int)
    u.loc[:, 'period'] = np.digitize(pickup_time.dt.hour, [0, 6, 18])
    u.loc[:, 'speed'] = speed(t)
    return u

train = augment(train)
test = augment(test)
train.iloc[0,:] # An example row
```

Out[35]:
```
pickup_datetime     2016-01-21 18:02:20
dropoff_datetime    2016-01-21 18:27:54
pickup_lon                      -73.9942
pickup_lat                       40.751
dropoff_lon                     -73.9637
dropoff_lat                      40.7711
passengers                             1
distance                            2.77
duration                            1534
date                          2016-01-21
hour                                  18
day                                    3
weekend                                0
period                                 3
speed                            6.50065
Name: 14043, dtype: object
```

4 Q2b **1 / 1**

✓ **+ 1 pts Correct**

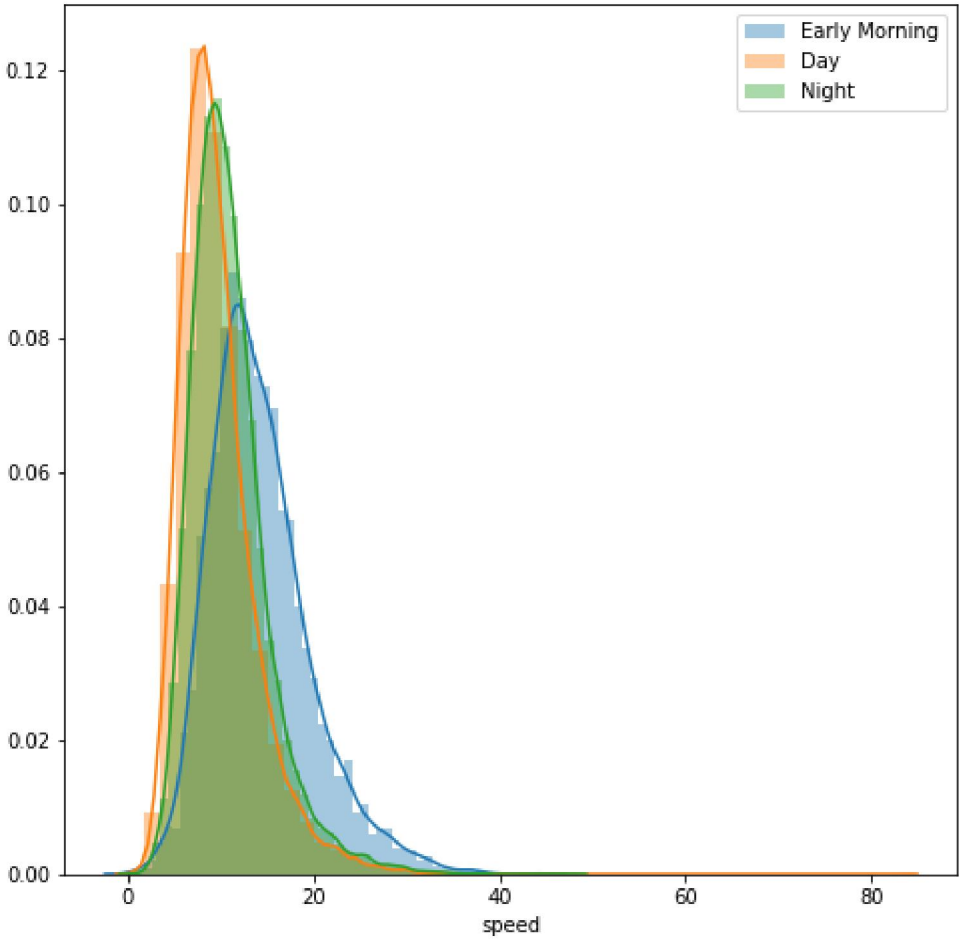
+ 0 pts Wrong

+ 0.5 pts partially correct

gradescope

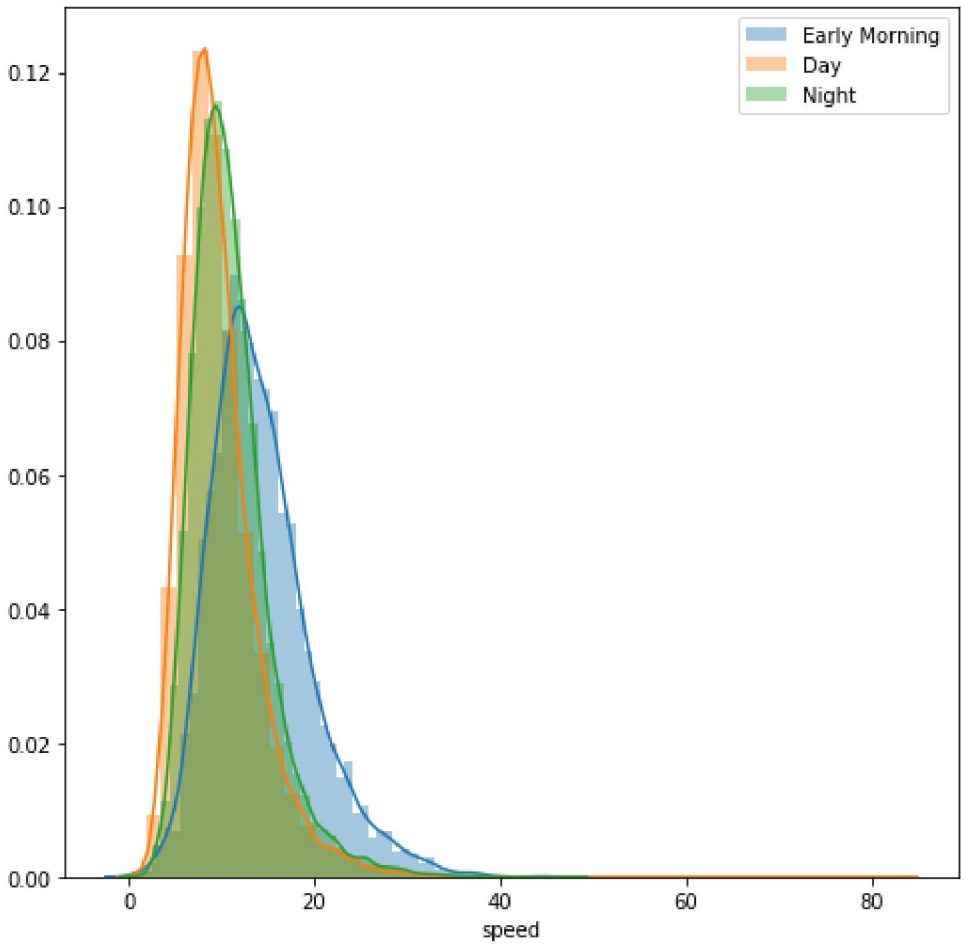
Question 2c

Use `sns.distplot` to create an overlaid histogram comparing the distribution of average speeds for taxi rides that start in the early morning (12am-6am), day (6am-6pm; 12 hours), and night (6pm-12am; 6 hours). Your plot should look like this:



In [36]:
```
plt.figure(figsize=(8, 8))
# BEGIN YOUR CODE
# ------------------------
plt.figure(figsize=(8, 8))
for i, s in enumerate(['Early Morning', 'Day', 'Night']):
    sns.distplot(train[train['period'] == i+1]['speed'], label=s)
plt.legend();
# ------------------------
# END YOUR CODE
```

```
<matplotlib.figure.Figure at 0x25f8a2b52e8>
```



It looks like the time of day is associated with the average speed of a taxi ride.

5 Q2c **2 / 2**

✓ **+ 2 pts Correct**

+ 0 pts Wrong

+ 1 pts partially correct

gradescope

Question 2d (PCA)

Manhattan can roughly be divided into Lower, Midtown, and Upper regions. Instead of studying a map, let's approximate by finding the first principal component of the pick-up location (latitude and longitude).

- Add a `region` column to `train` that categorizes each pick-up location as 0, 1, or 2 based on the value of each point's first principal component, such that an equal number of points fall into each region.
- Read the documentation of `pd.qcut` (https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.qcut.html), which categorizes points in a distribution into equal-frequency bins.
- You don't need to add any lines to this solution. Just fill in the assignment statements to complete the implementation.

The provided tests ensure that you have answered the question correctly.

In [37]:
```
# Find the first principle component
D = train[['pickup_lon', 'pickup_lat']].values
pca_n = D.shape[0]
pca_means = np.mean(D, axis=0)
X = (D - pca_means) / np.sqrt(pca_n)
u, s, vt = np.linalg.svd(X, full_matrices=False)

def add_region(t):
    """Add a region column to t based on vt above."""
    # BEGIN YOUR CODE
    # ------------------------
    D = t[['pickup_lon', 'pickup_lat']].values
    assert D.shape[0] == t.shape[0], 'You set D using the incorrect table'

    # Always use the same data transformation used to compute vt
    X = (D - pca_means) / np.sqrt(pca_n)
    first_pc = X @ vt.T[:,0] # SOLUTION
    # ------------------------
    # END YOUR CODE
    t.loc[:,'region'] = pd.qcut(first_pc, 3, labels=[0, 1, 2])

add_region(train)
add_region(test)
```

In [38]:
```
ok.grade("q2d");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------------
Test summary
    Passed: 7
    Failed: 0
[ooooooooook] 100.0% passed
```

Let's see how PCA divided the trips into three groups. These regions do roughly correspond to

- Lower Manhattan (below 14th street)
- Midtown Manhattan (between 14th and the park)
- Upper Manhattan (bordering Central Park).

No prior knowledge of New York geography was required!

6 Q2d 2 / 2

- ✓ **+ 2 pts** **Correct**
- **+ 0 pts** Wrong
- **+ 1.5 pts** partially correct 75%
- **+ 1 pts** partially correct 50%
- **+ 1.7 pts** partially correct 86%
- **+ 1.6 pts** partially correct 80%

gradescope

Part 3: Model Selection

In this part, you will select a regression model to predict the duration of a taxi ride.

Important: *Tests in this part do not confirm that you have answered correctly. Instead, they check that you're somewhat close in order to detect major errors. It is up to you to calculate the results correctly based on the question descriptions.*

Question 3a

Assign `constant_rmse` to the root mean squared error on the test set for a constant model that always predicts the mean duration of all training set taxi rides.

In [46]:
```
def rmse(errors):
    """Return the root mean squared error."""
    return np.sqrt(np.mean(errors ** 2))

# BEGIN YOUR CODE
# ------------------------
constant_rmse = rmse(np.mean(train['duration']) - test['duration'])
# ------------------------
# END YOUR CODE
constant_rmse
```

Out[46]:
```
399.14375723526661
```

In [47]:
```
ok.grade("q3a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

7 Q3a **2 / 2**

✓ **+ 2 pts Correct**

+ 0 pts Wrong

+ 1 pts partially correct 50%

gradescope

Question 3b

Assign `simple_rmse` to the root mean squared error on the test set for a simple linear regression model that uses only the distance of the taxi ride as a feature (and includes an intercept).

Terminology Note: Simple linear regression means that there is only one covariate. Multiple linear regression means that there is more than one. In either case, you can use the `LinearRegression` model from `sklearn` to fit

In [50]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
# BEGIN YOUR CODE
# ------------------------
model.fit(train[['distance']],train['duration'])
predictions =  model.predict(test[['distance']])
# ------------------------
# END YOUR CODE
errors = predictions - test['duration']
simple_rmse = rmse(errors)
simple_rmse
```

Out[50]: 276.78411050003422

In [51]:
```
ok.grade("q3b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

8 Q3b **2 / 2**

✓ **+ 2 pts Correct**

+ 0 pts Wrong

+ 1 pts partially correct

gradescope

Question 3c

Assign `linear_rmse` to the root mean squared error on the test set for a linear regression model fitted to the training set without regularization, using the design matrix defined by the `design_matrix` function from Part 3.

The provided tests check that you have answered the question correctly and that your `design_matrix` *function is working as intended.*

In [52]:
```
model = LinearRegression()
# BEGIN YOUR CODE
# ------------------------
model.fit(design_matrix(train), train['duration'])
predictions = model.predict(design_matrix(test))
# ------------------------
# END YOUR CODE
errors = predictions - test['duration']
linear_rmse = rmse(errors)
linear_rmse
```

Out[52]:
```
255.19146631882754
```

In [53]:
```
ok.grade("q3c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[oooooooooo] 100.0% passed
```

9 Q3c **2 / 2**

✓ **+ 2 pts Correct**

+ **0 pts** Wrong

+ **1.5 pts** partially correct 75%

+ **1.3 pts** partially correct 67%

+ **1 pts** partially correct 50%

gradescope

Question 3d

For each possible value of `period`, fit an unregularized linear regression model to the subset of the training set in that `period`. Assign `period_rmse` to the root mean squared error on the test set for a model that first chooses linear regression parameters based on the observed period of the taxi ride, then predicts the duration using those parameters. Again, fit to the training set and use the `design_matrix` function for features.

In [55]:
```
model = LinearRegression()
errors = []

for v in np.unique(train['period']):
    # BEGIN YOUR CODE
    # ------------------------
    v_train = train[train['period'] == v]
    v_test = test[test['period'] == v]
    model.fit(design_matrix(v_train), v_train['duration'])
    predictions = model.predict(design_matrix(v_test))
    # ------------------------
    # END YOUR CODE
    errors.extend(predictions - v_test['duration'])
period_rmse = rmse(np.array(errors))
period_rmse
```

Out[55]:
```
246.62868831165173
```

In [56]:
```
ok.grade("q3d");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

This approach is a simple form of decision tree regression, where a different regression function is estimated for each possible choice among a collection of choices. In this case, the depth of the tree is only 1.

10 Q3d **2 / 2**

✓ **+ 2 pts Correct**

+ 0 pts Wrong

+ 1 pts partially correct 50%

+ 1.3 pts partially correct 67%

gradescope

Question 3e

In one or two sentences, explain how the `period` regression model could possibly outperform linear regression when the design matrix for linear regression already includes one feature for each possible hour, which can be combined linearly to determine the `period` value.

Answer: The period regression model outperforms the linear regression model due to the fact that it divides the data up into "clusters" which have distinct patterns from each other. The ordinary regression model is less accurate because it is essentially averaging over all of these three clusters while the period model creates an individual model for each of the three different clusterings and thus can predict more accurately for each individual cluster.

Question 3f

Instead of predicting duration directly, an alternative is to predict the average *speed* of the taxi ride using linear regression, then compute an estimate of the duration from the predicted speed and observed distance for each ride.

Assign `speed_rmse` to the root mean squared error in the **duration** predicted by a model that first predicts speed as a linear combination of features from the `design_matrix` function, fitted on the training set, then predicts duration from the predicted speed and observed distance.

Hint: Speed is in miles per hour, but duration is measured in seconds. You'll need the fact that there are 60 * 60 = 3,600 seconds in an hour.

11 Q3e **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts Wrong

gradescope

Question 3e

In one or two sentences, explain how the `period` regression model could possibly outperform linear regression when the design matrix for linear regression already includes one feature for each possible hour, which can be combined linearly to determine the `period` value.

Answer: The period regression model outperforms the linear regression model due to the fact that it divides the data up into "clusters" which have distinct patterns from each other. The ordinary regression model is less accurate because it is essentially averaging over all of these three clusters while the period model creates an individual model for each of the three different clusterings and thus can predict more accurately for each individual cluster.

Question 3f

Instead of predicting duration directly, an alternative is to predict the average *speed* of the taxi ride using linear regression, then compute an estimate of the duration from the predicted speed and observed distance for each ride.

Assign `speed_rmse` to the root mean squared error in the **duration** predicted by a model that first predicts speed as a linear combination of features from the `design_matrix` function, fitted on the training set, then predicts duration from the predicted speed and observed distance.

Hint: Speed is in miles per hour, but duration is measured in seconds. You'll need the fact that there are 60 * 60 = 3,600 seconds in an hour.

In [60]:
```
model = LinearRegression()
# BEGIN YOUR CODE
# ------------------------
model.fit(design_matrix(train), train['speed'])
speed_predictions = model.predict(design_matrix(test))
duration_predictions = test['distance']/ speed_predictions * 60 * 60
# ------------------------
# END YOUR CODE
errors = duration_predictions - test['duration']
speed_rmse = rmse(errors)
speed_rmse
```

Out[60]: 243.01798368514949

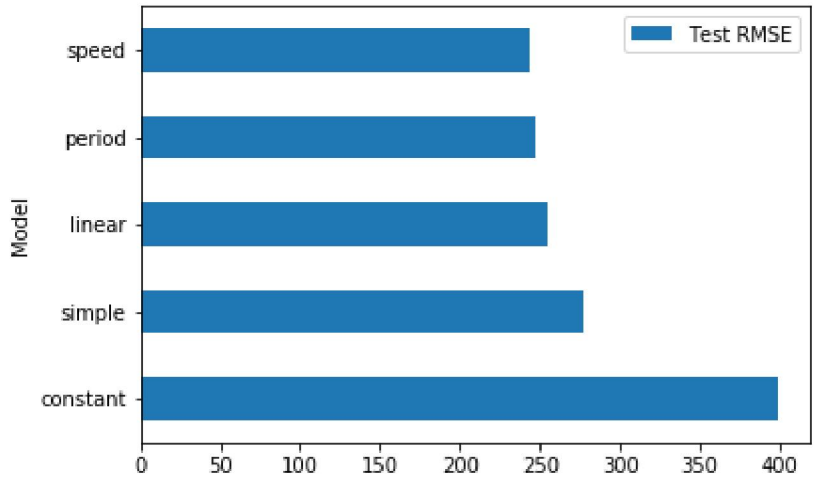
In [61]:
```
ok.grade("q3f");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Here's a summary of your results:

In [62]:
```
models = ['constant', 'simple', 'linear', 'period', 'speed']
pd.DataFrame.from_dict({
    'Model': models,
    'Test RMSE': [eval(m + '_rmse') for m in models]
}).set_index('Model').plot(kind='barh');
```



12 Q3f **2 / 2**

✓ **+ 2 pts Correct**

+ 0 pts Wrong

+ 1.5 pts partially correct 75%

+ 1 pts partially correct 50%

gradescope