CSE519_HW2

September 25, 2018

In [243]:
```
import pandas
import datetime
import numpy as np
import geopy.distance
import os
from IPython.display import display, HTML
import random
import seaborn
from matplotlib import pyplot as plt
from matplotlib import cm
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import linear_model
from sklearn import svm
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.ensemble import GradientBoostingRegressor
```

In [9]:
```
def get_distance(a, b):
    return geopy.distance.distance(a, b).miles

#https://stackoverflow.com/questions/40807225/how-to-call-data-from-a-dataframe-into-h
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)

    All args must be of equal length.

    """
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2
```

```
            c = 2 * np.arcsin(np.sqrt(a))
            km = 6367 * c
            return km

In [10]: def pre_process(chunk):
             chunk['distance'] = chunk[['pickup_latitude',
                                        'pickup_longitude',
                                        'dropoff_latitude',
                                        'dropoff_longitude']
                                       ].apply(lambda x: get_distance(x[0], x[1], x[2], x[3]), a

             chunk.drop(chunk[ chunk['distance'] == 0 ].index, inplace = True)

             write_chunk(chunk, "cleaned_data.csv")

In [11]: def clean(chunk):
             newyork_lat = 40.730610
             newyork_long = -73.935242
             range_var = 0.5
             chunk.dropna(inplace=True)
             chunk.drop(chunk[chunk['fare_amount'] <= 0.0].index, inplace = True)
             chunk.drop(chunk[chunk['passenger_count'] == 0].index, inplace = True)
             chunk.drop(chunk[(chunk['pickup_latitude'] < newyork_lat-range_var) | (chunk['pick
             chunk.drop(chunk[(chunk['dropoff_latitude'] < newyork_lat-range_var) | (chunk['dro
             chunk.drop(chunk[(chunk['pickup_longitude'] < newyork_long-range_var) | (chunk['pi
             chunk.drop(chunk[(chunk['dropoff_longitude'] < newyork_long-range_var) | (chunk['d

In [12]: def write_chunk(chunk, filename):
             chunk.to_csv(filename, columns = ['fare_amount','passenger_count', 'distance', 'ti

In [13]: chunk_size = 10**7
         count = 0
         if os.path.exists('cleaned_data.csv'):
             os.remove('cleaned_data.csv')

         chunk = pandas.read_csv('all/train.csv', nrows = chunk_size)
         count += 1
         #display(chunk.describe())
         print("\rProcessing count: {}".format(count * chunk_size), end="")

Processing count: 10000000

In [14]: clean(chunk)

In [15]: pre_process(chunk)

In [18]: chunk['time_hour'] = pandas.to_datetime(
                 chunk['pickup_datetime'].apply(lambda x: x.split()[1]),
                 format='%H:%M:%S').dt.hour
```

```
In [19]: chunk['year'] = pandas.to_datetime(
                 chunk['pickup_datetime'].apply(lambda x: x.split()[0]),
                 format='%Y-%m-%d'
             ).dt.year.astype(int)

In [20]: chunk['month'] = pandas.to_datetime(
                 chunk['pickup_datetime'].apply(lambda x: x.split()[0]),
                 format='%Y-%m-%d'
             ).dt.month.astype(int)
```

Pearsons Coefficient

```
In [119]: percentage = 0.001
          random_rows = pandas.read_csv('cleaned_data.csv',
                                        dtype={
                                             'fare_amount': np.float32,
                                             'distance': np.float32,
                                             'passenger_count': int,
                                             'time_hour': int,
                                            'year' : int
                                         },
                                          skiprows=lambda i: i>0 and random.random() > percentage

          random_rows['time_part_of_day'] = pandas.cut(random_rows['time_hour'],
                                                       [-1, 6, 12, 16 ,24],
                                                       labels = [1, 2, 3 ,4]).astype(int)
          random_rows = random_rows.filter(['fare_amount', 'passenger_count', 'distance', 'time
          display(random_rows.head())
          random_rows.describe()
```

	fare_amount	passenger_count	distance	time_hour	year	time_part_of_day
0	7.7	1	1.737223	4	2012	1
1	6.5	1	1.109777	11	2013	2
2	4.5	1	0.811588	17	2012	4
3	21.5	1	4.734918	23	2013	4
4	16.9	2	3.950192	3	2012	1

Out[119]:

	fare_amount	passenger_count	distance	time_hour	year	\
count	9646.000000	9646.000000	9646.000000	9646.000000	9646.000000	
mean	11.321514	1.699150	2.090501	13.427016	2011.724238	
std	9.420152	1.312497	2.215263	6.494773	1.869692	
min	2.500000	1.000000	0.000148	0.000000	2009.000000	
25%	6.100000	1.000000	0.808063	9.000000	2010.000000	
50%	8.500000	1.000000	1.356755	14.000000	2012.000000	
75%	12.500000	2.000000	2.484034	19.000000	2013.000000	
max	124.500000	6.000000	17.819082	23.000000	2015.000000	

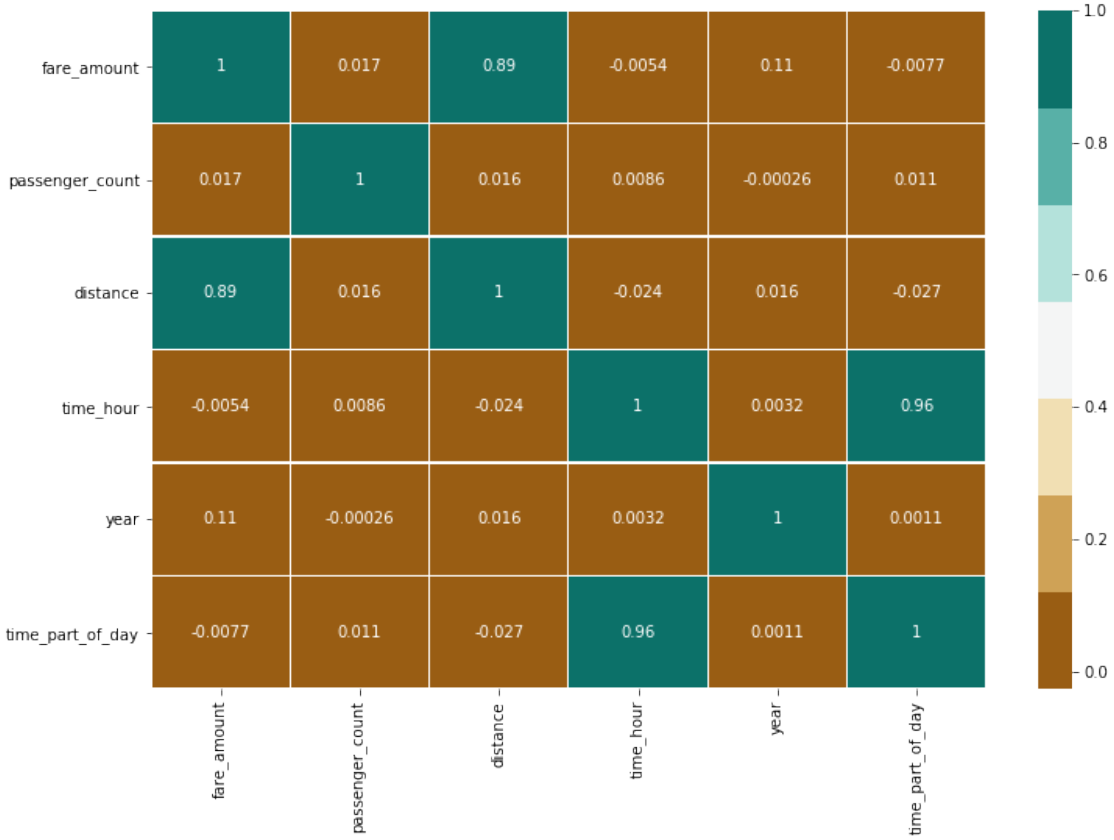
	time_part_of_day

```
count       9646.000000
mean           2.814120
std            1.102884
min            1.000000
25%            2.000000
50%            3.000000
75%            4.000000
max            4.000000
```

In [120]:
```
# Reference: https://stackoverflow.com/questions/22258491/read-a-small-random-sample
```

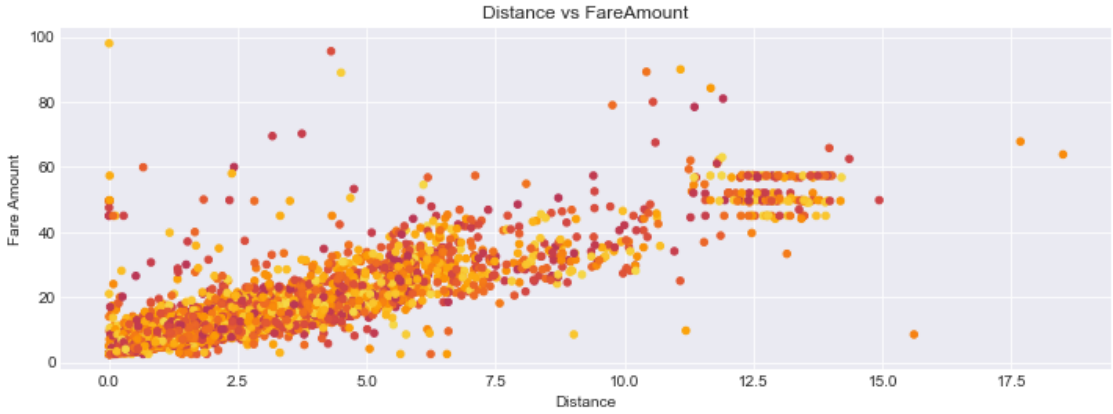
In [121]:
```
pearson_corr = random_rows[['fare_amount', 'passenger_count', 'distance', 'time_hour
display(pearson_corr)
labels = ['fare_amount', 'passenger_count', 'distance', 'time_hour',
                'year', 'time_part_of_day']
seaborn.heatmap(pearson_corr,
                cmap=seaborn.color_palette("BrBG", 7),
            linewidths=.1,
            annot=True,
            xticklabels=labels,
            yticklabels=labels)
plt.gcf().set_size_inches(12,8)
```

```
                  fare_amount  passenger_count  distance  time_hour      year  \
fare_amount          1.000000         0.017381  0.890199  -0.005405  0.113453
passenger_count      0.017381         1.000000  0.016091   0.008638 -0.000265
distance             0.890199         0.016091  1.000000  -0.023727  0.016486
time_hour           -0.005405         0.008638 -0.023727   1.000000  0.003175
year                 0.113453        -0.000265  0.016486   0.003175  1.000000
time_part_of_day    -0.007684         0.010713 -0.026650   0.958932  0.001134

                  time_part_of_day
fare_amount              -0.007684
passenger_count           0.010713
distance                 -0.026650
time_hour                 0.958932
year                      0.001134
time_part_of_day          1.000000
```



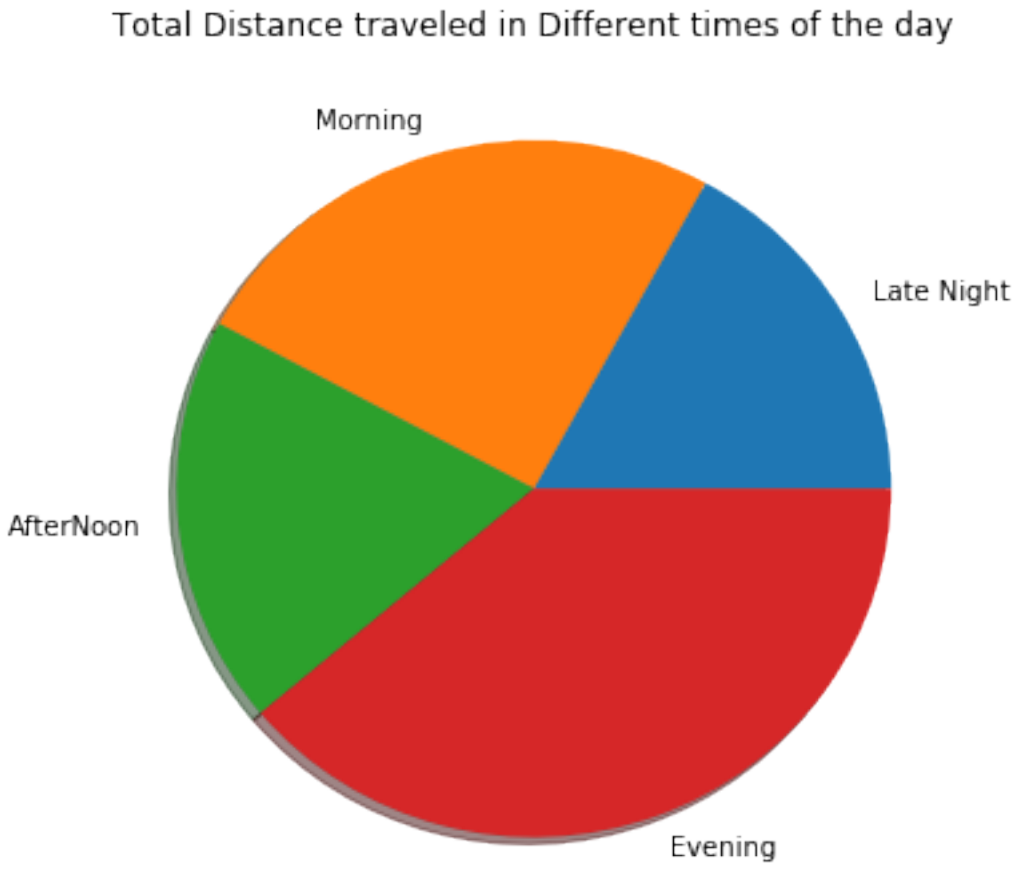
Visualization

```
In [6]: seaborn.set_style("darkgrid")
        plot = random_rows[['fare_amount', 'distance']].plot(kind='scatter',
                                                             x = 'distance',
                                                             y = 'fare_amount',
                                                             figsize=(12,4),
                                                             color = cm.inferno_r(np.linspace(.1,.5,
                                                             title="Distance vs FareAmount")


        plot.set_xlabel("Distance")
        _ = plot.set_ylabel("Fare Amount")
```



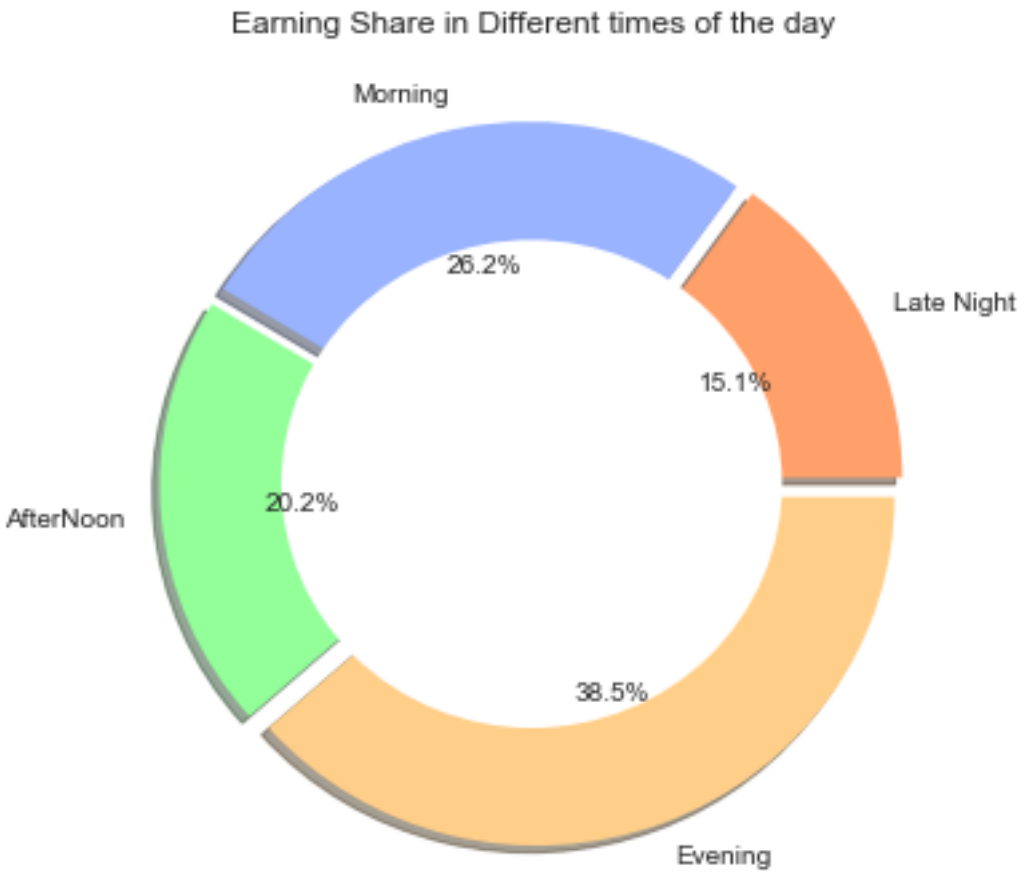
Reference: https://medium.com/@kvnamipara/a-better-visualisation-of-pie-charts-by-matplotlib-935b7667d77f

```
In [3]: sum_fareamount = random_rows.groupby('time_part_of_day')['distance'].sum()
        plot = sum_fareamount.plot(kind ='pie',
                                   x = 'time_hour',
                                   y = 'distance',
                                   title='Total Distance traveled in Different times of the day',
                                   legend = False,
                                   figsize=(6,6), shadow=True, labels=["Late Night", 'Morning', 'Afte
        plot.set_xlabel('')
        _ = plot.set_ylabel('')
```

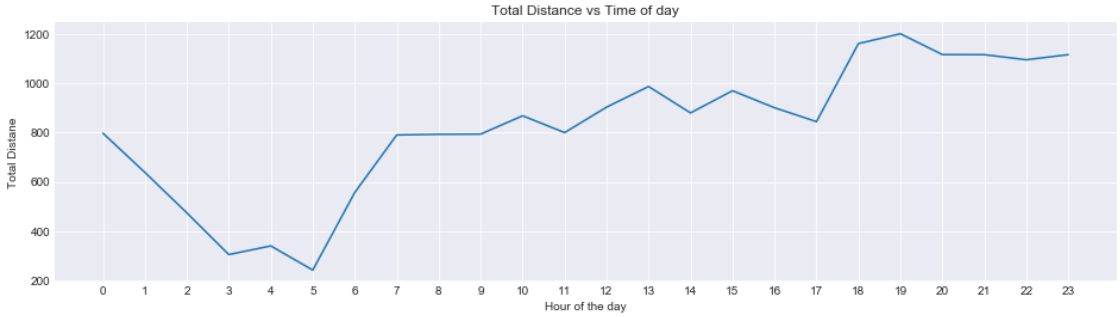



```
In [7]: sum_fareamount = random_rows.groupby('time_part_of_day')['fare_amount'].sum()
        plot = sum_fareamount.plot(kind ='pie',
                                   x = 'time_hour',
                                   y = 'fare_amount',
                                   title='Earning Share in Different times of the day',
                                   legend = False,
                                   figsize=(6,6),
                                   shadow=True,
                                   labels=["Late Night", 'Morning', 'AfterNoon', 'Evening'],
                                   colors = ['#ff9f69','#99b3ff','#92ff99','#ffcf89'],
                                   explode = (0.04, 0.04, 0.04, .04),
                                   autopct='%1.1f%%')

        centre_circle = plt.Circle((0,0),0.70, fc = 'white')
        plt.gcf().gca().add_artist(centre_circle)

        plot.set_xlabel('')
        plot.set_ylabel('')
```

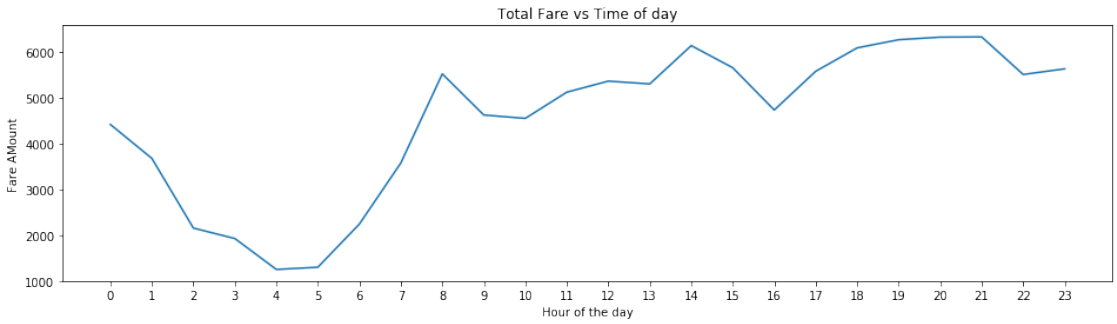
Out[7]: Text(0,0.5,'')



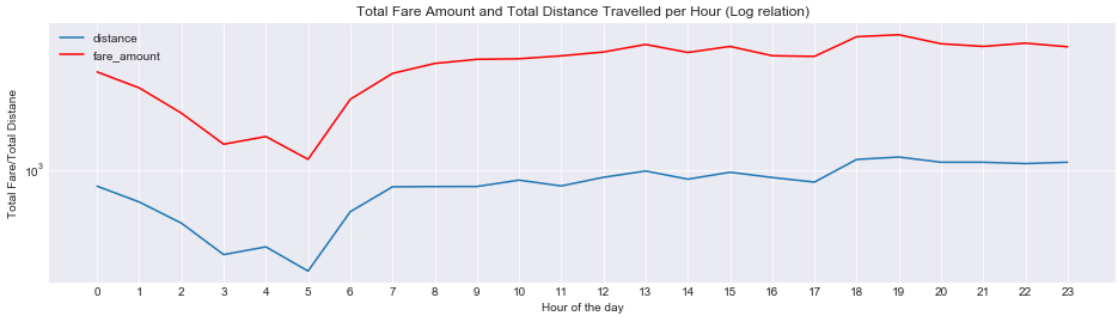
In [8]:
```
sum_distance = random_rows.groupby('time_hour')['distance'].sum()
plot = sum_distance.plot(kind ='line',
                         x = 'time_hour',
                         y = 'distance',
                         title='Total Distance vs Time of day',
                         figsize=(16,4))
plot.set_xlabel("Hour of the day")
plot.set_ylabel("Total Distane")
_ = plot.set_xticks(range(0,24,1))
```



In [286]:
```
sum_fare = random_rows.groupby('time_hour')['fare_amount'].sum()
plot = sum_fare.plot(kind ='line',
                     x = 'time_hour',
                     y = 'fare_amount',
                     title='Total Fare vs Time of day',
                     figsize=(16,4))
plot.set_xlabel("Hour of the day")
plot.set_ylabel("Fare AMount")
_ = plot.set_xticks(range(0,24,1))
```



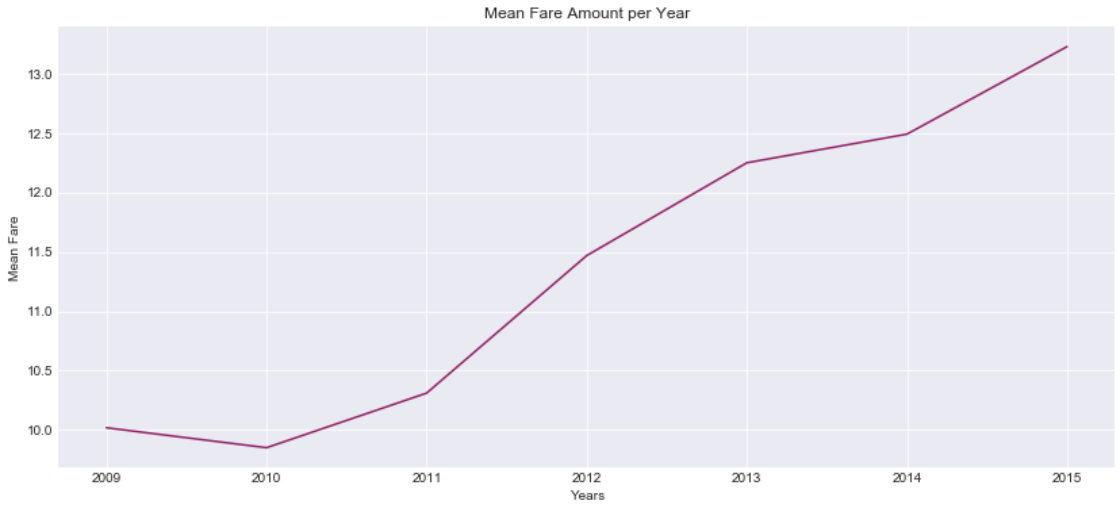
In [9]:
```
sum_distance = random_rows.groupby('time_hour')['distance'].sum()
sum_fareamount = random_rows.groupby('time_hour')['fare_amount'].sum()

sum_distance.plot(kind ='line', x = 'time_hour', y = 'distance', legend = True)
plot = sum_fareamount.plot(kind ='line', x = 'time_hour', y = 'fare_amount',
                     figsize=(16,4),
                     title="Total Fare Amount and Total Distance Travelled per Hour (Log
                     legend = True,
                     logy= True, color='Red')
plot.set_xlabel("Hour of the day")
plot.set_ylabel("Total Fare/Total Distane")
_ = plot.set_xticks(range(0,24,1))
```



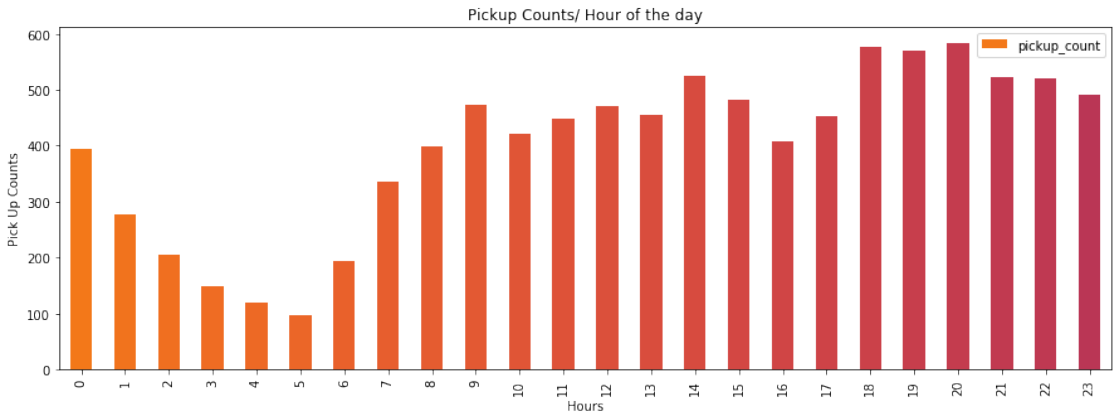
In [10]:
```
#Reference :  https://stackoverflow.com/questions/11927715/how-to-give-a-pandas-matpl
```

In [11]:
```
year_fare_mean = pandas.DataFrame(random_rows.groupby('year')['fare_amount'].mean().as
                                  pandas.Series(range(2009,2016), name="year"))

    display(year_fare_mean)
    plot = year_fare_mean.plot(kind = 'line',
                   stacked=True,
                   color = cm.inferno_r(np.linspace(.6,.9, 24)),
                   legend=False,
                   figsize=(14,6),
                   title="Mean Fare Amount per Year")
    plot.set_xlabel("Years")
    _ = plot.set_ylabel("Mean Fare")
```

year	fare_amount
2009	10.017088
2010	9.848972
2011	10.309377
2012	11.468890
2013	12.251040
2014	12.492615
2015	13.230117




```
In [25]: sum_pickup = pandas.DataFrame()
         sum_pickup['time_hour'] = pandas.Series(range(0,24))
         sum_pickup['pickup_count'] = random_rows.time_hour.value_counts().astype(np.int)
         sum_pickup = sum_pickup.filter(['pickup_count', 'time_hour'])
         plot = sum_pickup.plot(kind = 'bar',
                        x = 'time_hour',
                        y = 'pickup_count',
                        stacked=True,
                        color = cm.inferno_r(np.linspace(.3,.5, 24)),
                        figsize=(15,5),
                        title="Pickup Counts/ Hour of the day" )
         plot.set_xlabel("Hours")
         _ = plot.set_ylabel("Pick Up Counts")
```



New generated Features

```
In [291]: cleaned_csv = pandas.read_csv('cleaned_data.csv')
          cleaned_csv['time_part_of_day'] = pandas.cut(cleaned_csv['time_hour'],
                                                       [-1, 6, 12, 16 ,24],
                                                       labels = [1, 2, 3, 4]).astype(int)
          feature_csv = pandas.merge(cleaned_csv,
                                     sum_pickup[['time_hour','pickup_count']],
                                     on='time_hour')

          feature_csv = feature_csv.filter(['fare_amount',
                                      'passenger_count',
                                      'distance',
                                      'time_hour',
                                      'year',
                                      'time_part_of_day',
                                      'pickup_count',
                                      ], axis=1)

          display(feature_csv.head())
```

	fare_amount	passenger_count	distance	time_hour	year	time_part_of_day	\
0	4.5	1	0.639764	17	2009	4	
1	16.5	1	2.584861	17	2012	4	
2	7.7	2	1.037366	17	2011	4	
3	7.5	1	1.007130	17	2012	4	
4	13.5	1	1.997993	17	2013	4	

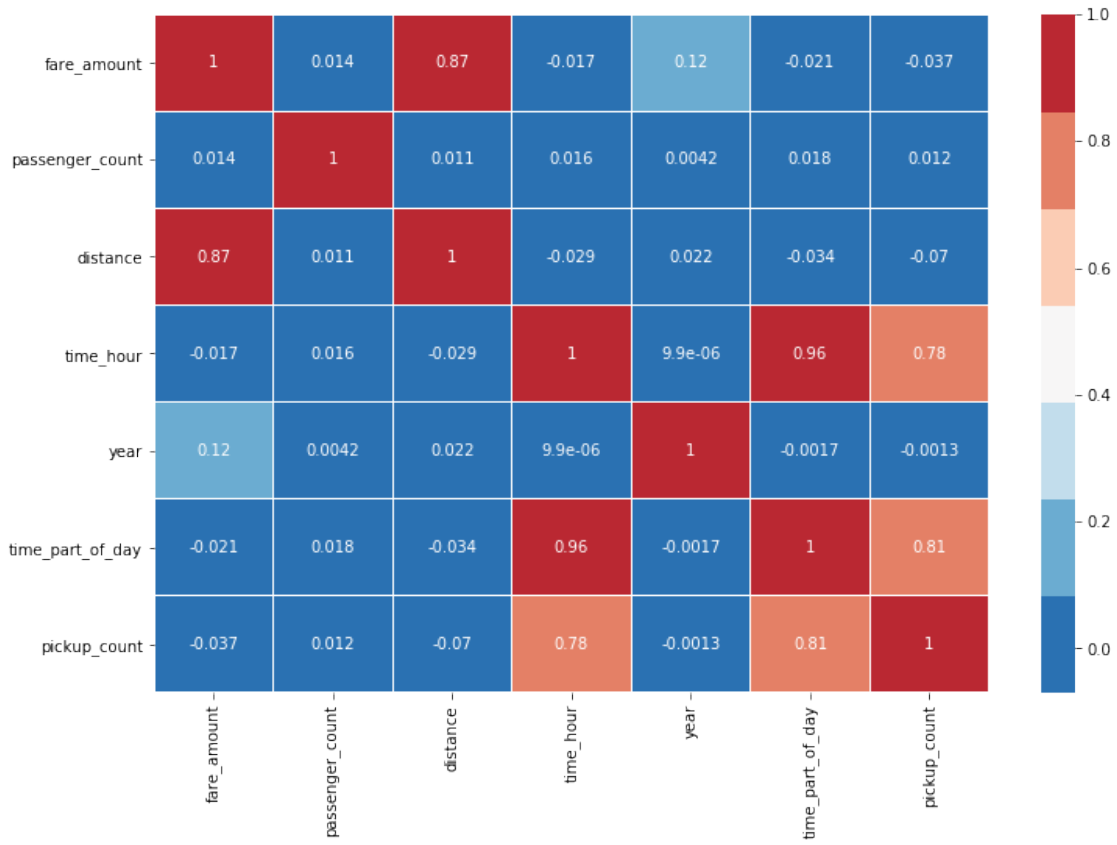
	pickup_count
0	452
1	452
2	452
3	452
4	452

Correlation Heatmap of the new features

```
In [292]: pearson_corr = feature_csv.corr(method='pearson')
          display(feature_csv.head())
          labels = [[ 'fare_amount', 'passenger_count', 'distance', 'time_hour',
                      'year', 'time_part_of_day', 'pickup_count']]

          seaborn.heatmap(pearson_corr,
                          cmap=seaborn.color_palette("RdBu_r", 7),
                      linewidths=.1,
                      annot=True)
          plt.gcf().set_size_inches(12,8)
```

	fare_amount	passenger_count	distance	time_hour	year	time_part_of_day	\
0	4.5	1	0.639764	17	2009	4	

```
1          16.5                1  2.584861          17  2012                4
2           7.7                2  1.037366          17  2011                4
3           7.5                1  1.007130          17  2012                4
4          13.5                1  1.997993          17  2013                4

   pickup_count
0           452
1           452
2           452
3           452
4           452
```



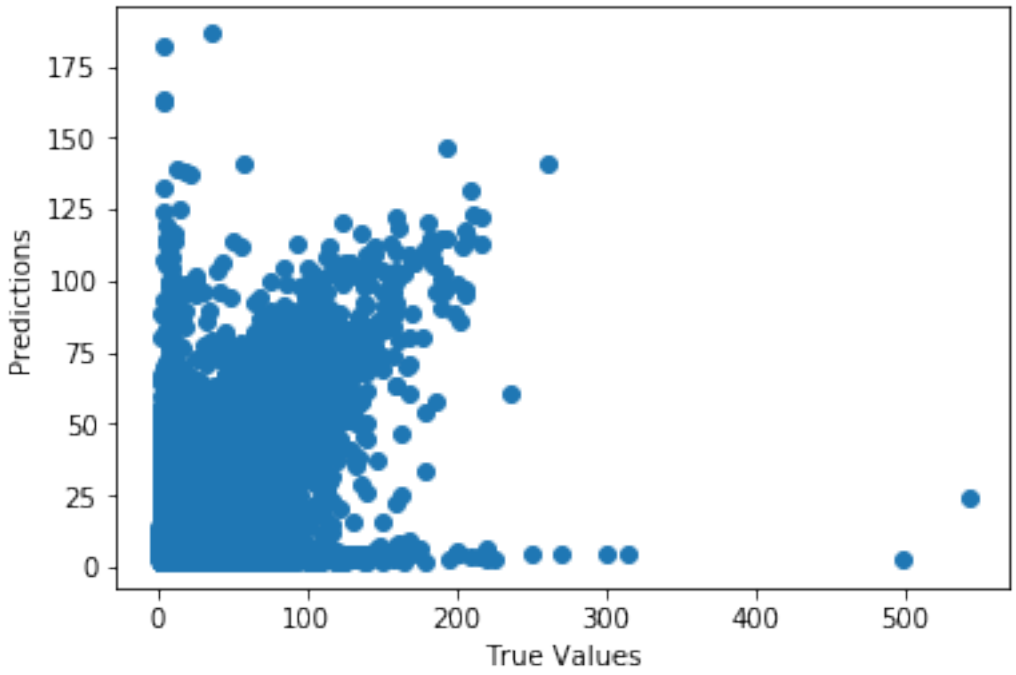
Baseline model: Linear Regression

```
In [293]: X = feature_csv[['distance', 'time_part_of_day', 'year']]
          Y = feature_csv[['fare_amount']]

In [294]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
          print(X_train.shape, Y_train.shape)

(7717943, 3) (7717943, 1)
```

```
In [295]: baseline = linear_model.LinearRegression()
          baseline_model = baseline.fit(X_train, Y_train)
          predictions = baseline.predict(X_test)
```

Generate Baseline Model Scores

```
In [296]: plt.scatter(Y_test, predictions)
          plt.xlabel("True Values")
          plt.ylabel("Predictions")

Out[296]: Text(0,0.5,'Predictions')
```




```
In [297]: print("Accuracy: "+str((baseline_model.score(X_test, Y_test)*100).round())+"%",
                "\tRMS Error: " + str(mean_squared_error(Y_test, predictions)))

Accuracy: 77.0%         RMS Error: 20.936110128423508
```

Advanced model: Using Random Forest Regressor Reference: https://stackoverflow.com/questions/41925157/logisticregression-unknown-label-type-continuous-using-sklearn-in-python

```
In [225]: #X = feature_csv[['passenger_count', 'distance', 'time_hour', 'time_part_of_day', 'p
          feature_csv = feature_csv[(feature_csv.fare_amount / feature_csv.distance) >= 1]
          feature_csv = feature_csv[(feature_csv.fare_amount / feature_csv.distance) <= 23]
          X = feature_csv[['distance', 'year', 'time_hour']]
```