Kapil Bastola
Lab 9

1. Complete reference to the IEEE or ACM paper you are trying to outperform. Only articles published by IEEE or ACM will be accepted.

   Lippi, M., Bertini, M., & Frasconi, P. (2013). Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning. IEEE Transactions on Intelligent Transportation Systems, 14(2), 871-882. doi:10.1109/tits.2013.2247040

2. Describe the results of the paper you have chosen.

   The competitors have presented an extensive experimental review of many statistical and machine-learning approaches to short-term traffic flow forecasting in the paper. Following the approach in SARIMA, they have also proposed two new SVR models, employing a seasonal kernel to measure similarity between time-series examples. The presented results confirm that seasonality is a key feature in achieving high accuracy; however, the most accurate models often require high computational resources both during the training phase and at prediction time. For this reason, the presented seasonal kernel approach might be a reasonable compromise between forecasting accuracy and computational complexity issues. The SARIMA version that does not include a Kalman filter and the ANNs perform consistently worse than SVR with an RBF kernel, which, in turn, is less accurate than the seasonal- kernel variant. As a future direction of research, we conjecture that a significant improvement in time-series forecasting may come from relational learning, where interrelations between different time series are taken into account. In a road network, for example, the traffic conditions in neighboring nodes are inherently interdependent, and exploiting this interdependence should be crucial in improving the prediction accuracy of related time series. Moreover, different from classical statistical approaches, relational learning algorithms can easily deal with multiple sources of information, which might become a crucial feature in the forthcoming years, where huge data sets might become available from wireless sensors, Global Positioning System, and floating-car data. Another very interesting direction of research, which has been indicated by the experimental results presented in this paper, consists of investigating the covariate shift in traffic

3. Describe your results so far, and your next objective in terms of data analysis.

   I have looked at the descriptive statistics of the datasets using various aggregation operators like mean, median, etc. Next I will apply different algorithms to my dataset and compare the different methods as well as compare them with the competitor's methods.

4. Describe how you will use the concepts of aggregation and group operations

   I will use aggregation and group operations to look at the descriptive statistics of my data. It will be helpful to understand what the mean average speed of traffic is by hour of the day to see if there is any difference between average speeds by hour of day. The same analysis can be done with vehicle count.

5. Create a table showing the code for every operation in the left column and the time measurements for every operation in the right column. You need at least 10 operations. Highlight in bold the lines of code where you use aggregation or group operation concepts.

Code	Time
roadtraffic = pd.concat(map(pd.read_csv, glob.glob(os.path.join("/Users/Kapil/Downloads/traffic_feb_june", "*.csv"))))	48 sec
%pyspark **roadtraffic.count()**	4 secs
%pyspark import re roadtraffic['hour'] = roadtraffic['TIMESTAMP'].str[11:13]	5 secs
grouped_avgspeed_byhour = **roadtraffic['avgSpeed'].groupby(roadtraffic['hour'])** **print(grouped.mean())**	**<1 sec**
grouped_vehicleCount_byhour = **roadtraffic['vehicleCount'].groupby(roadtraffic['hour'])** **grouped_vehicleCount_byhour.mean()**	<1 sec
%sql select count(_id) from road_traffic	10 secs
%pyspark close_px = pd.read_csv('/Users/Kapil/Downloads/stock_px.csv', parse_dates=True, index_col=0) close_px[-4:]	<1 sec

%pyspark **# Annual correlation of Apple with Microsoft** **by_year.apply(lambda g: g['AAPL'].corr(g['MSFT']))**	<1 sec
%pyspark **import statsmodels.api as sm** **def regression(data, yvar, xvars):** **Y = data[yvar]** **X = data[xvars]** **X['intercept'] = 1.** **result = sm.OLS(Y,X).fit()** **return result.params** **by_year.apply(regression,'AAPL',['SPX'])**	3 secs
%sql select * from road_traffic limit 15 --see the first 15 rows in the table	1 sec