Prediction of Bike Share Demand

-EL-GY 9123 Introduction to Machine Learning Final Project

Hongyi Sun(hs3497) Qiongxuan Tan(qt310)

Date: 05/16/2018

Project Abstract:

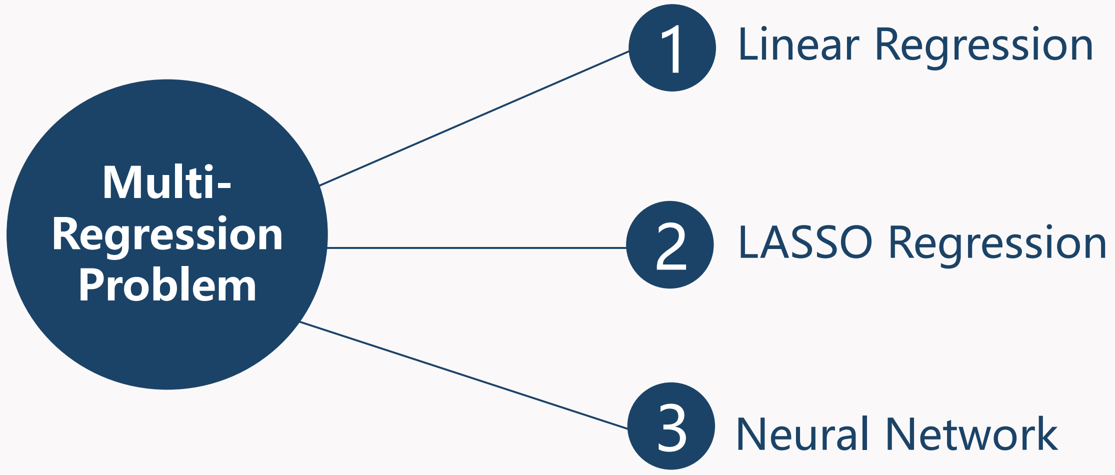
- With the popularity of shared bike in big cities like San Francisco, it is becoming more and more important to predict the real demand of shared bike based on previous data.

  We try to predict the daily demand of shared bike and duration of each trip based on several weather and date features.

  Since there are several features and targets, we consider it as a multi-regression problem. We got data from Kaggle.com[1] and try to fit it in models we learned in class, such as linear regression, LASSO regression, and neural network. We compare the accuracies of different models to get the optimized solution. After getting different accuracies from these models, we will discuss how to improve it in future work

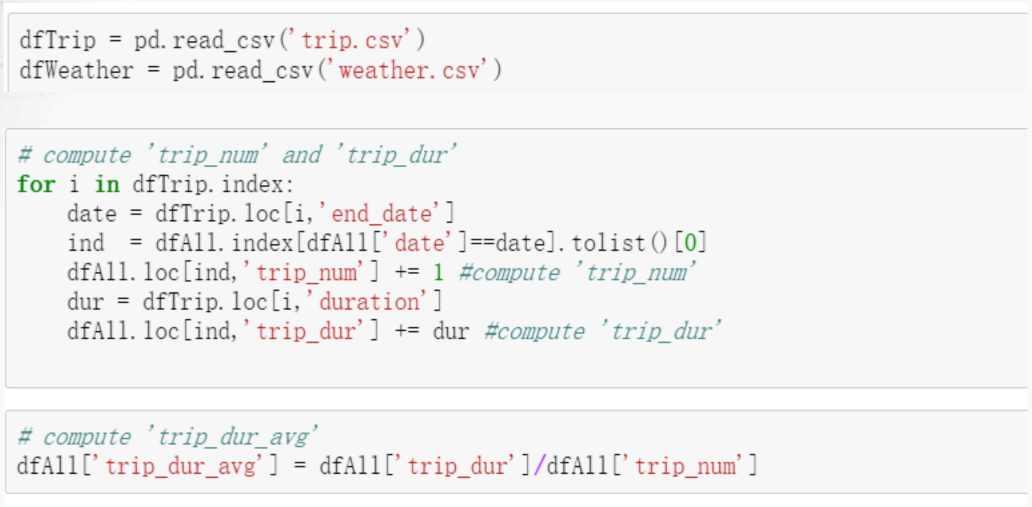
[1] https://www.kaggle.com/benhamner/sf-bay-area-bike-share

Project Abstract:



Build Dataset:

Selection of features and targets



```
dfTrip = pd.read_csv(’trip.csv’)
dfWeather = pd.read_csv(’weather.csv’)
```

```
# compute ’trip_num’ and ’trip_dur’
for i in dfTrip.index:
    date = dfTrip.loc[i,’end_date’]
    ind  = dfAll.index[dfAll[’date’]==date].tolist()[0]
    dfAll.loc[ind,’trip_num’] += 1 #compute ’trip_num’
    dur = dfTrip.loc[i,’duration’]
    dfAll.loc[ind,’trip_dur’] += dur #compute ’trip_dur’
```

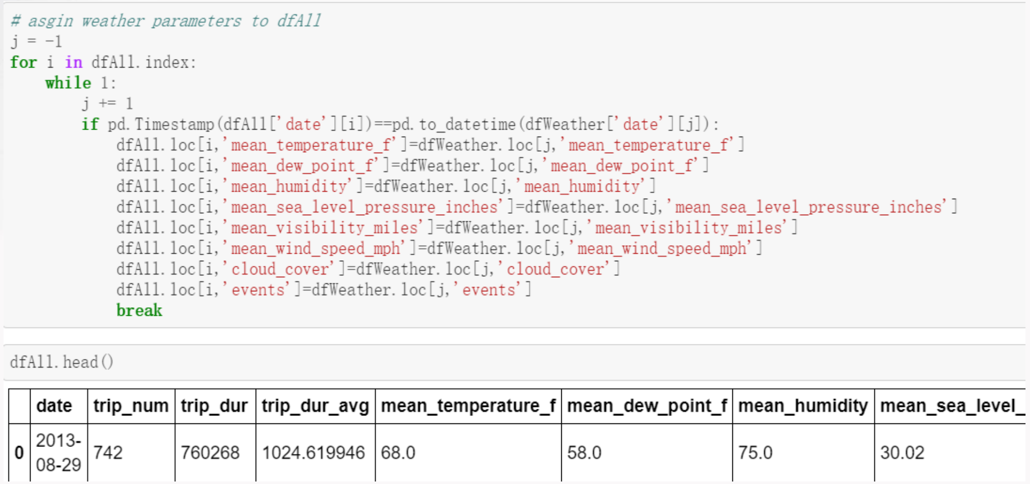
```
# compute ’trip_dur_avg’
dfAll[’trip_dur_avg’] = dfAll[’trip_dur’]/dfAll[’trip_num’]
```

1. Read data from trip.csv and weather.csv.

2. Compute the daily trip numbers and average durations. In this step, we explore end date of each trip, count all trips with same end date as daily trip number, and use mean of their durations as average trip duration.

Build Dataset:

Selection of features and targets

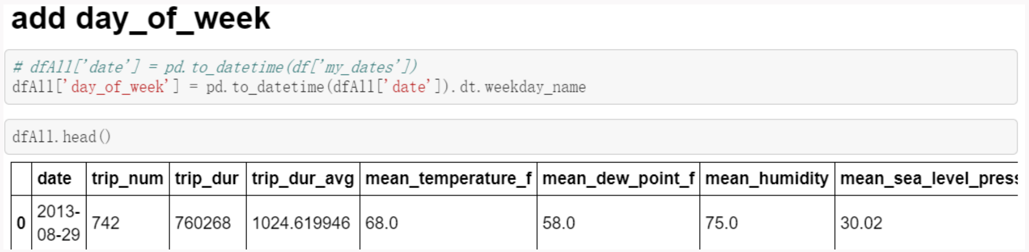


```
# asgin weather parameters to dfAll
j = -1
for i in dfAll.index:
    while 1:
        j += 1
        if pd.Timestamp(dfAll['date'][i])==pd.to_datetime(dfWeather['date'][j]):
            dfAll.loc[i,'mean_temperature_f']=dfWeather.loc[j,'mean_temperature_f']
            dfAll.loc[i,'mean_dew_point_f']=dfWeather.loc[j,'mean_dew_point_f']
            dfAll.loc[i,'mean_humidity']=dfWeather.loc[j,'mean_humidity']
            dfAll.loc[i,'mean_sea_level_pressure_inches']=dfWeather.loc[j,'mean_sea_level_pressure_inches']
            dfAll.loc[i,'mean_visibility_miles']=dfWeather.loc[j,'mean_visibility_miles']
            dfAll.loc[i,'mean_wind_speed_mph']=dfWeather.loc[j,'mean_wind_speed_mph']
            dfAll.loc[i,'cloud_cover']=dfWeather.loc[j,'cloud_cover']
            dfAll.loc[i,'events']=dfWeather.loc[j,'events']
            break
```

```
dfAll.head()
```

	date	trip_num	trip_dur	trip_dur_avg	mean_temperature_f	mean_dew_point_f	mean_humidity	mean_sea_level_
0	2013-08-29	742	760268	1024.619946	68.0	58.0	75.0	30.02

3. Match weather parameters with daily trip parameters. In this step, we choose weather features like temperature, dew point, humidity, sea level, visibility, wind speed, cloud cover and events



add day_of_week

```
# dfAll['date'] = pd.to_datetime(df['my_dates'])
dfAll['day_of_week'] = pd.to_datetime(dfAll['date']).dt.weekday_name
```

```
dfAll.head()
```

	date	trip_num	trip_dur	trip_dur_avg	mean_temperature_f	mean_dew_point_f	mean_humidity	mean_sea_level_press
0	2013-08-29	742	760268	1024.619946	68.0	58.0	75.0	30.02

4. We also add the day of week feature by common sense since it is supposed to have difference demands between workdays and weekends.

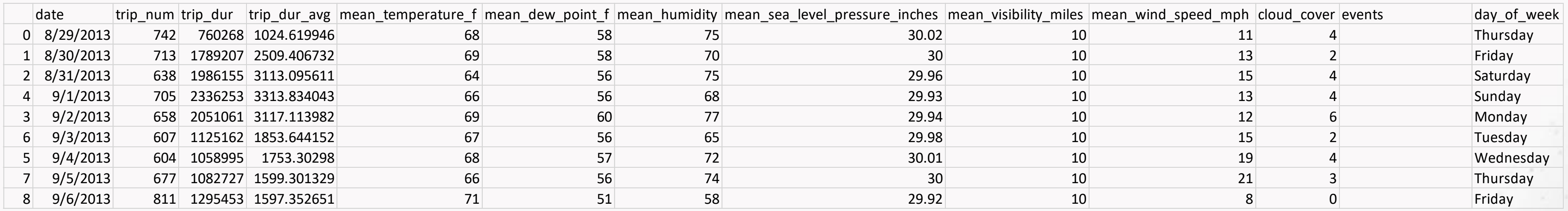
Build Dataset:

Selection of features and targets

Features:

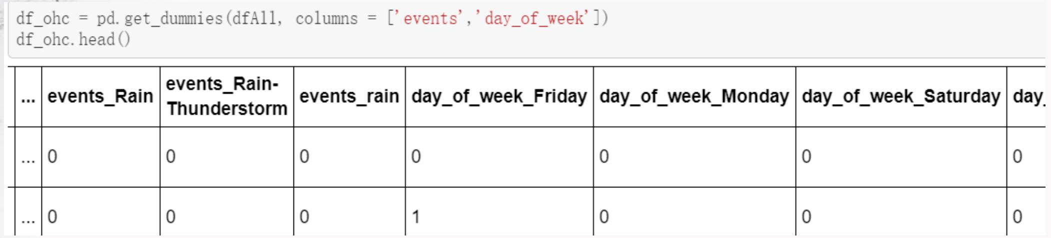
date, mean_temperature_f, mean_dew_point_f, mean_humidity, mean_sea_level_pressure_inches, mean_visibility_miles, mean_wind_speed_mph, cloud_cover, events

Targets: trip_num, trip_dur_avg



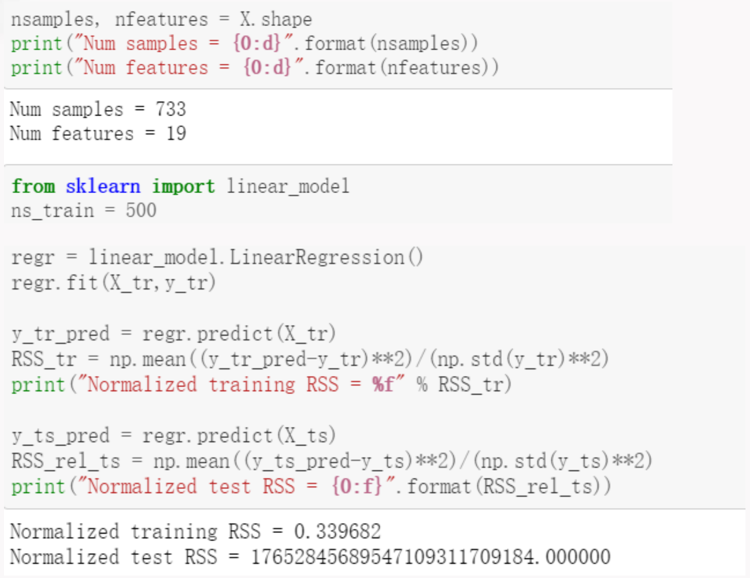
	date	trip_num	trip_dur	trip_dur_avg	mean_temperature_f	mean_dew_point_f	mean_humidity	mean_sea_level_pressure_inches	mean_visibility_miles	mean_wind_speed_mph	cloud_cover	events	day_of_week
0	8/29/2013	742	760268	1024.619946	68	58	75	30.02	10	11	4		Thursday
1	8/30/2013	713	1789207	2509.406732	69	58	70	30	10	13	2		Friday
2	8/31/2013	638	1986155	3113.095611	64	56	75	29.96	10	15	4		Saturday
4	9/1/2013	705	2336253	3313.834043	66	56	68	29.93	10	13	4		Sunday
3	9/2/2013	658	2051061	3117.113982	69	60	77	29.94	10	12	6		Monday
6	9/3/2013	607	1125162	1853.644152	67	56	65	29.98	10	15	2		Tuesday
5	9/4/2013	604	1058995	1753.30298	68	57	72	30.01	10	19	4		Wednesday
7	9/5/2013	677	1082727	1599.301329	66	56	74	30	10	21	3		Thursday
8	9/6/2013	811	1295453	1597.352651	71	51	58	29.92	10	8	0		Friday

Linear Regression:



```
df_ohc = pd.get_dummies(dfAll, columns = ['events','day_of_week'])
df_ohc.head()
```

...	events_Rain	events_Rain-Thunderstorm	events_rain	day_of_week_Friday	day_of_week_Monday	day_of_week_Saturday	day_
...	0	0	0	0	0	0	0
...	0	0	0	1	0	0	0



```
nsamples, nfeatures = X.shape
print("Num samples = {0:d}".format(nsamples))
print("Num features = {0:d}".format(nfeatures))
```

```
Num samples = 733
Num features = 19
```

```
from sklearn import linear_model
ns_train = 500
```

```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)

y_tr_pred = regr.predict(X_tr)
RSS_tr = np.mean((y_tr_pred-y_tr)**2)/(np.std(y_tr)**2)
print("Normalized training RSS = %f" % RSS_tr)

y_ts_pred = regr.predict(X_ts)
RSS_rel_ts = np.mean((y_ts_pred-y_ts)**2)/(np.std(y_ts)**2)
print("Normalized test RSS = {0:f}".format(RSS_rel_ts))
```

```
Normalized training RSS = 0.339682
Normalized test RSS = 17652845689547109311709184.000000
```

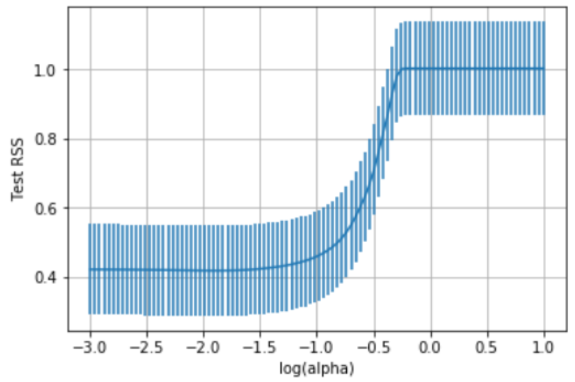
1. Use one hot coding in feature ‘events’ and ‘day of week’, set number of training samples to 500 and the rest for test.
2. Even though the normalized training RSS is small, the linear regression model still fails because the normalized test RSS is extremely large, this is because overfitting due to large number of parameters and relatively small number of samples, which lead to large errors in test dataset.
3. Therefore, we try to use LASSO regression to select optimal features.

LASSO Regression:

```
import sklearn.model_selection
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold, shuffle=True)
nalpha = 100
alpha_test = np.logspace(-3, 1, nalpha)
model = sklearn.linear_model.Lasso(warm_start=True)
RSSts = np.zeros((nalpha, nfold))
```

```
RSS_mean = np.mean(RSSts,  axis=1)
RSS_std = np.std(RSSts,  axis=1)/np.sqrt(nfold)
plt.errorbar(np.log10(alpha_test), RSS_mean, yerr=RSS_std)
plt.grid()
plt.xlabel('log(alpha)')
plt.ylabel('Test RSS')
```

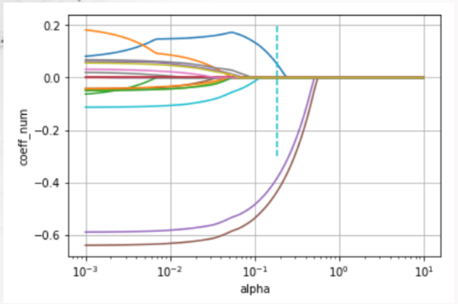
```
Text(0, 0.5, 'Test RSS')
```

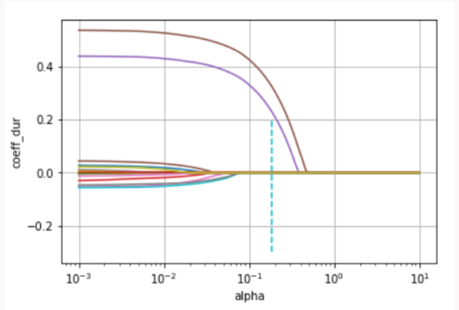


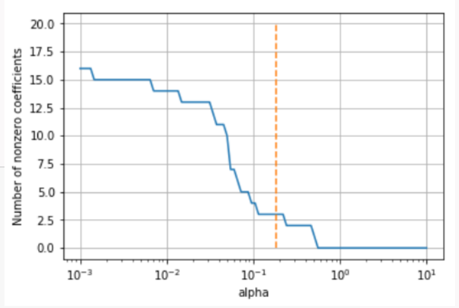
```
The optimal alpha=   1.8307e-01
Mean test RSS = 0.530085
```

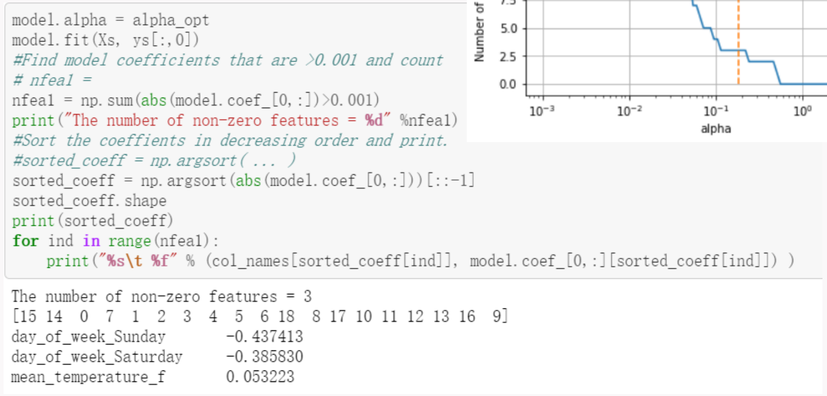
1. Use cross validation to select the regularization level alpha. Choose 100 alpha values logarithmically spaced from 1e-3 to 10 and 10 fold for cross validation. Store the test RSS in a matrix RSSts with rows correponding to different alpha values and columns corresponding to different cross validation folds.
2. Determine the RSS mean and standard error corresponding to different alpha and plot the mean RSS with error bar as a function of alpha.
3. Find the optimal alpha and the mean test RSS at this optimal point using the one standard error rule.
4. The mean test RSS is 0.530085 after LASSO regularization, which is a reasonable result.

LASSO Regression:









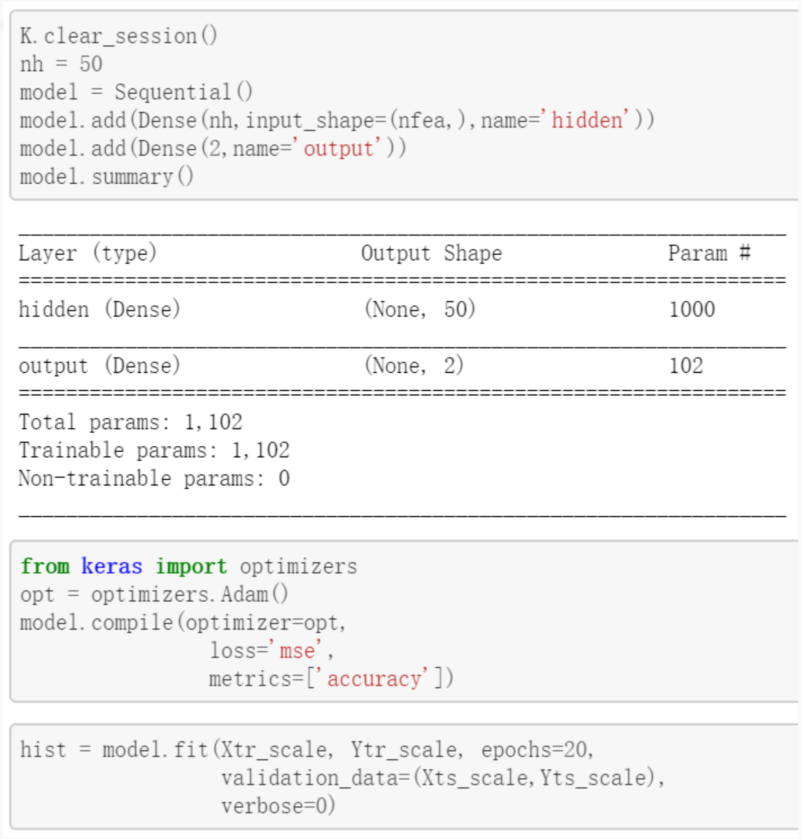
```
model.alpha = alpha_opt
model.fit(Xs, ys[:,0])
#Find model coefficients that are >0.001 and count
# nfeal =
nfeal = np.sum(abs(model.coef_[0,:])>0.001)
print("The number of non-zero features = %d" %nfeal)
#Sort the coeffients in decreasing order and print.
#sorted_coeff = np.argsort( ... )
sorted_coeff = np.argsort(abs(model.coef_[0,:]))[::-1]
sorted_coeff.shape
print(sorted_coeff)
for ind in range(nfeal):
    print("%s\t %f" % (col_names[sorted_coeff[ind]], model.coef_[0,:][sorted_coeff[ind]]) )
```

```
The number of non-zero features = 3
[15 14  0  7  1  2  3  4  5  6 18  8 17 10 11 12 13 16  9]
day_of_week_Sunday       -0.437413
day_of_week_Saturday     -0.385830
mean_temperature_f       0.053223
```

5. We also draw the LASSO path for both trip numbers and trip durations to illustration the effect of LASSO regularization, we can see from the figure that features with large coefficients are the same.
6. Find the number of non-zero LASSO coefficients, which is 3, and the corresponding feature names are day_of_week_Sunday, day_of_week_Saturday and mean_temperature_f.
7. These coefficients actually make sense in real life.

Neural Network:

Build a 2 layers Neural Network



```
K.clear_session()
nh = 50
model = Sequential()
model.add(Dense(nh, input_shape=(nfea,), name='hidden'))
model.add(Dense(2, name='output'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 50)                1000
_________________________________________________________________
output (Dense)               (None, 2)                 102
=================================================================
Total params: 1,102
Trainable params: 1,102
Non-trainable params: 0
_________________________________________________________________
```

```
from keras import optimizers
opt = optimizers.Adam()
model.compile(optimizer=opt,
                loss='mse',
                metrics=['accuracy'])
```

```
hist = model.fit(Xtr_scale, Ytr_scale, epochs=20,
                 validation_data=(Xts_scale,Yts_scale),
                 verbose=0)
```

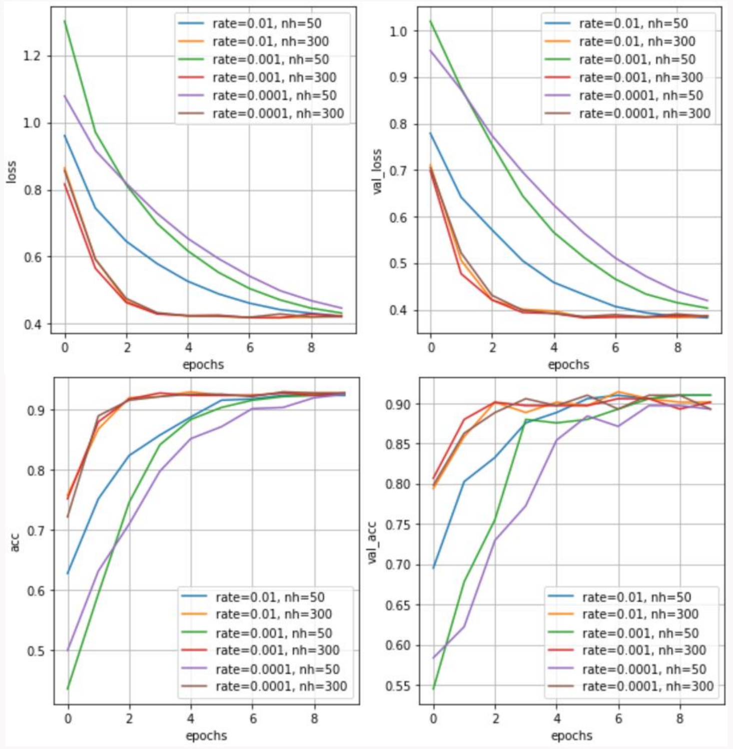
Regression case: Multi-Regression Problem

Loss function:

$$MSE = \frac{1}{NK}\sum_{i=1}^{N}\sum_{j=1}^{K}(y_{ik} - z_{Oik})^2$$

Neural Network:

Optimizing parameters



To optimize the result of a 2 layer neural network, we trained the network using different learning rates and numbers of hidden units listed as follow:

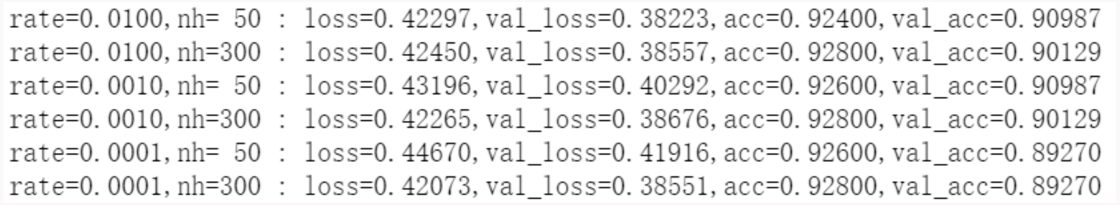
Learning rates(rate): 0.01, 0.001, 0.0001

Lumbers of hidden units(nh): 50, 300

From the resulting losses and accuracies, it is obvious that using large number of hidden units converges faster. When using 300 hidden units, the best learning rate is 0.001.

Neural Network:

Final results



```
rate=0.0100, nh= 50  :  loss=0.42297, val_loss=0.38223, acc=0.92400, val_acc=0.90987
rate=0.0100, nh=300  :  loss=0.42450, val_loss=0.38557, acc=0.92800, val_acc=0.90129
rate=0.0010, nh= 50  :  loss=0.43196, val_loss=0.40292, acc=0.92600, val_acc=0.90987
rate=0.0010, nh=300  :  loss=0.42265, val_loss=0.38676, acc=0.92800, val_acc=0.90129
rate=0.0001, nh= 50  :  loss=0.44670, val_loss=0.41916, acc=0.92600, val_acc=0.89270
rate=0.0001, nh=300  :  loss=0.42073, val_loss=0.38551, acc=0.92800, val_acc=0.89270
```

The final results using different parameters are similar. The best accuracy is 92.80%, validation accuracy is 90.13%, with learning rate 0.01 and number of hidden units 300.

Conclusion and Future Work:

We performed linear regression, LASSO regression and neural network regression to predict the demand for bikes at bike share stations, as a function of day of week, temperature, and wind speed and so on.

In future work, we can predict the numbers and duration of bike trips as a function of age and gender of customer. Also, for weather data, we could divide it into 24 hour periods, and the demand of shared bike may have relationship with daytime and nighttime. It would also be interesting to model trips between pairs of stations, and build models to predict the destination given customer information and starting station.

THANKS