FireCallsGeorgina

December 2, 2019

```
In [1]: from MyFunctions import *
```

1 Fire calls to the San Francisco Fire Department (SFFD)

This project generates a model to forecast the next day number of non-medical calls to the San Francisco Fire Department, allowing the Department to get ready with the necessary resources and personnel to take care of the calls.

The calls could be related to fire, rescues, biological-hazards, explotions, industrial accidents, etc.

Information comes from different sources: - **Calls to SFFD:** More than 500k records with details about every call since 2003 (DataSF : Downloaded in July 28th 2019). - **Weather conditions:** NOAA - Daily information for precipitation, minimum and maximum temperature from SF downtown station - Daily information for: average wind and gusts (2mins and 5 secs) from SF airport

Data from the calls to the SFFD was aggregated by day and merged with weather data. Creation of the time series model involved **feature engineering** like: - **RainxGusts =** (precipitation x gust at 5 seconds)2 - **Moving holidays** such as thanks giving

The project was developped in Python 3 using libraries such as: Pandas, Sklearn and ipywidgets

1.1 Use this tool to forecast the number of calls to the San Francisco Fire Department:

Please enter your weather forecast using the boxes provided below, and hit the orange button to get the forecast. The last available day in our system is displayed. Forecast for the following day will be computed.

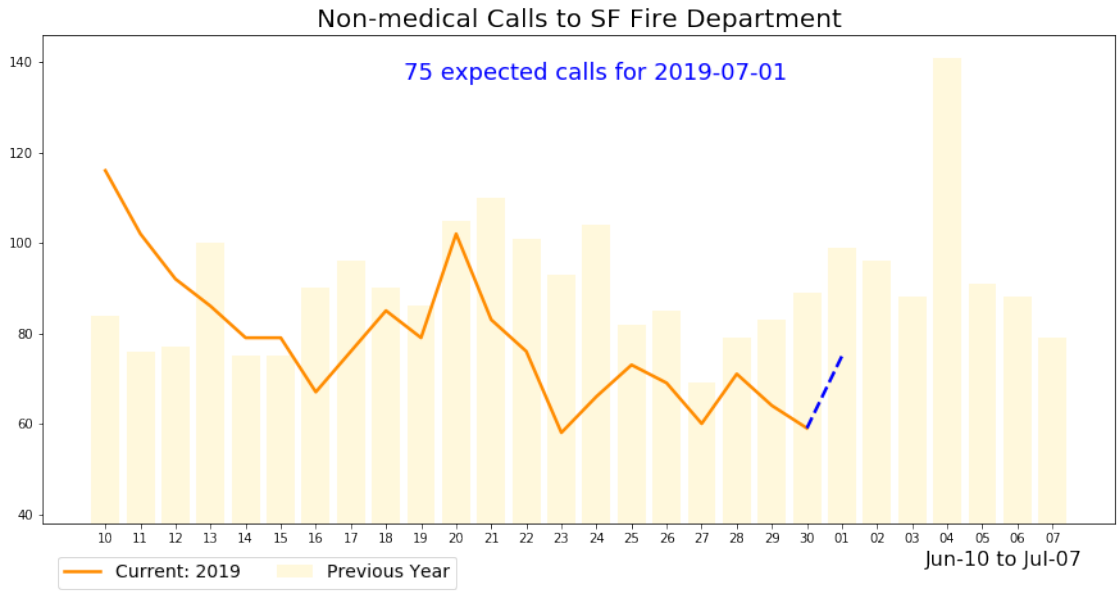
Note: To get another forecast for a different weather input, run the following cell again and then hit the forecast button.

```
In [2]: display_form()
```

```
Last available day in our database:  2019-06-30
Please enter weather forecast for  2019-07-01 :
```

```
HBox(children=(VBox(children=(BoundedFloatText(value=12.75, description='Average wind speed (MPH
```

```
Your input:
Date: 2019-07-01 AWND: [12.75] WSF5: [29.1] PRCP: [0.0] TMIN: [54.0] TMAX: [64.0]
```



1.2 Performance of the model

Data

- Trainig data: Jan 1st, 2003 to Dec 31st, 2016
- Validation data: Jan 1st, 2017 to Jun 30th, 2019

```
In [10]: DF = read_data2()
         DF_complete, indeps = add_features(DF)
         model, train, test = final_model(DF_complete)
```

Naive model Using the calls of the previous day to forecast the number of calls will provide us with a base model to compare the performance of the final model. Mean square error (MSE) is 391, and R2 score is negative (-0.38). Run the cell, and see the results for yourself.

```
In [11]: print('MSE Naive prev day= ', round(sklearn.metrics.mean_squared_error(test['Calls'], t
         print('R2 Naive prev day= ',round(sklearn.metrics.r2_score(test['Calls'], test['Calls_l
```

```
MSE Naive prev day=  391.0
R2 Naive prev day=  -0.38
```

Final model Different linear and non-linear models were tested including Random Forest and Ridge Linear Regression but all of them arrived to a similar score. The final model uses Linear Regression with an R2 score of 0.29 reducing the MSE error, from 391 with the naive model, to 200.

As a first step, Fourier analysis was used finding relevants cycles for: year, semester, quarters , months and weeks. Values for Fourier were included as part of the features in the final model which considers 54 variables including as well: the engineered featured of (precipitation x gusts at 5 secs)2, average wind, maximum temperature for the day, minimum temperature of the day, a smoothed value from the previous week (minimum between the calls from the previous week -same day- and 2 weeks before), volume of calls from the previous day, average from the previous year and 4 weeks moving average. In addition to those variables, it also uses indicator variables for: thanksgiving, black friday, weekday (Monday to Thursday), weekend (Friday and Saturday), Sunday and month.

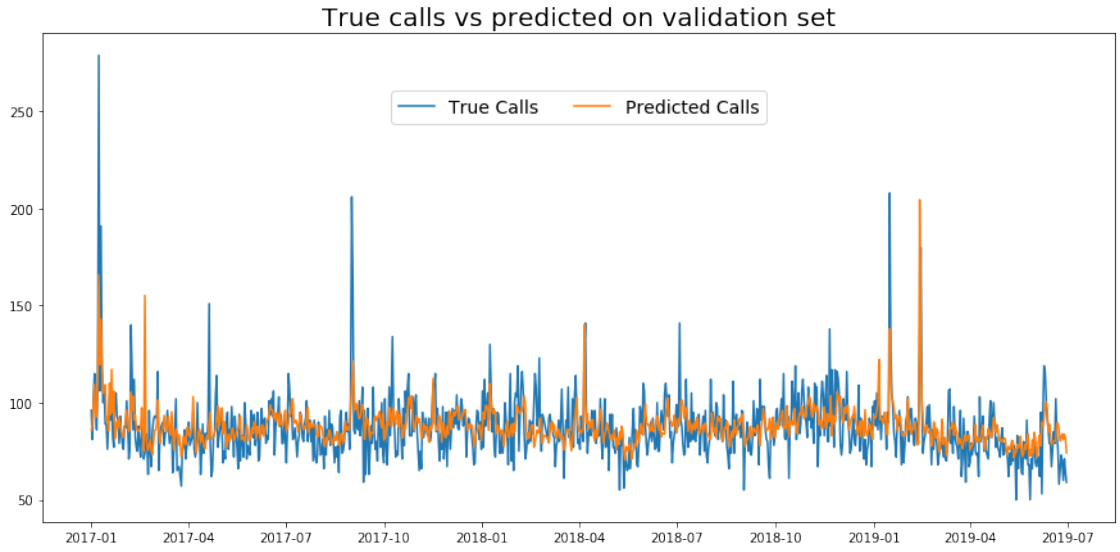
Although there is room for improvement, the final model follows well the trend and is able to predict some of the edge values. Some more feature engineering can be done given more time (including many relevant missing holidays and important dates like school calendar). Modeling noise might also be considered.

Run the following cell to see how forecasted values (orange) follow the original values (blue) for the volume of calls to SFFD:

Note: A model with almost the same performance an only 12 variables can also be seen under the section "Additional information for the curious ones: An heuristic approach"

```
In [12]: predicted = model.predict(X=test.drop(['Calls'],axis=1))
         metrics2('Test',test, 'Calls', predicted)

MSE final model:  200.0
R2 final model:  0.29
```



Scores A normalized version of the model is below allowing a fair comparisons of the coefficients. If you want to see the coefficients, run the next 3 cells:

```
In [13]: model_normalized = sklearn.linear_model.LinearRegression(normalize=True).fit(
                X=train.drop(['Calls'],axis=1), y=train['Calls'])

In [14]: indeps = set(test.columns)
         indeps.remove('Calls')
         coef_li = [(feat, round(coef,2)) for feat, coef in zip(list(indeps),model_normalized.co
```

If you want to take a peek, these are the coeficients in descending order of importance:

```
In [15]: def sortSec(val):
             return abs(val[1])

         # sort descending by 2nd element
         coef_li.sort(key = sortSec, reverse = True)
         print(coef_li)
```

```
[('smoothed_prev_week', -22600867069640.6), ('Friday', 7210144316896.69), ('Dec', 7210144316895.
```

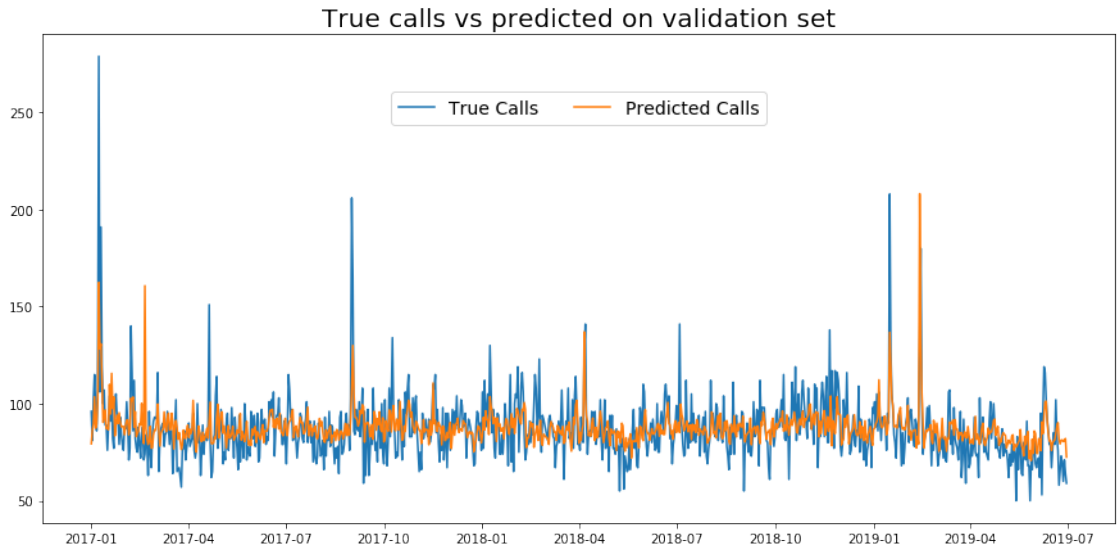
1.3 Additional information for the curious ones

1.3.1 An heuristic approach

By taking some of the most relevant features, it is possible to achieve almost the same performance with only 12 features (MSE=205, R2=0.28). These are: the engineered featured of (precipitation x gusts at 5 secs)2, maximum temperature for the day, a smoothed value from the previous week (minimum between the calls from the previous week -same day- and 2 weeks before), volume of calls from the previous day and average from the previous year. In addition to those variables, it also uses indicator variables for: thanksgiving, black friday, weekday (Monday to Thursday), weekend (Friday and Saturday), Sunday and the month of January. Run the next cell to see the performance of this simpler model:

```
In [16]: myFeatures = ['RainxGusts','TMAX','smoothed_prev_week','Calls_last_1d','Thanks_Thu','Th
                       'Weekend_FS','prev_year','Sunday','year_avg_1d','Jan']
         my_lr_model = sklearn.linear_model.LinearRegression().fit(
                X=train[myFeatures], y=train['Calls'])
         my_lr_predicted = my_lr_model.predict(X=test[myFeatures])
         metrics2('Test',test, 'Calls', my_lr_predicted)
```

```
MSE final model:  205.0
R2 final model:  0.28
```



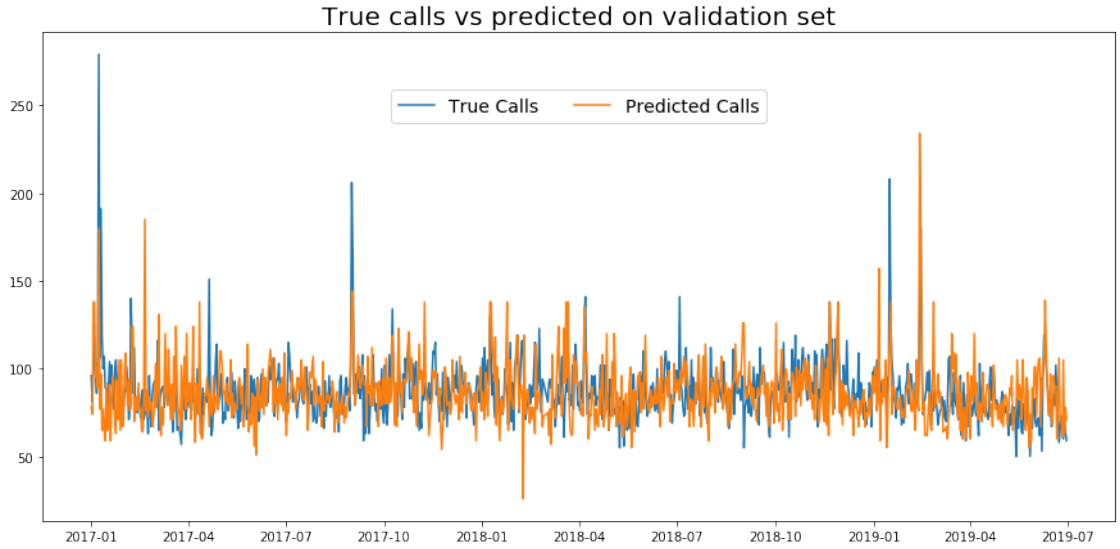

```
In [ ]: my_model_normalized = sklearn.linear_model.LinearRegression(normalize=True).fit(
                X=train[myFeatures], y=train['Calls'])
```

1.3.2 Decision trees instead of linear regression

Decision trees are a good alternative to explore non linear models. Considering that we already found 12 strong variables, a decision tree could provide us as well with a self-explanatory model. However, it does not perform well. The mean square error is larger than the one from the Naive model.

```
In [22]: my_tree_model = sklearn.tree.DecisionTreeRegressor(random_state=0).fit(X=train[myFeatur
         my_tree_predicted = my_tree_model.predict(X=test[myFeatures])
         metrics2('Test',test, 'Calls', my_tree_predicted)

MSE final model:  393.0
R2 final model:  -0.39
```



Random forest instead of linear regression Random forest achieved almost the same performance than linear regression. In addition to that, the final model is less intuitive. You can see the results by running the cell.

```
In [18]: my_forest_model= sklearn.ensemble.RandomForestRegressor(n_estimators=20, max_depth=200,
         my_forest_predicted = my_forest_model.predict(X=test[myFeatures])
         metrics2('Test',test, 'Calls', my_forest_predicted)
```

```
MSE final model:  220.0
R2 final model:  0.22
```