Exception Prediction Using Random Forest

Data Wrangling

Features	Details
EXCEPTION_HOURS	EXCEPTION_HOURS
NOTICE	= SHIFT_START_DATE_TIME - EXCEPTION_CREATION_DATE, in hours
JOB_FAMILY	DC1000, DC2B00,DC2A00
MONTH	Natural Month
WEEKDAY	if SHIFT_DATE is weekday, if TRUE, then 1, if FALSE, then 0
SITE	St Paul's Hospital, Mt St Joseph, Holy Family, SVH Langara, Brock Fahrni, Youville Residence

EARNING_CATEGORY	Original EARNING_CATEGORY	Number of Exceptions
Relief Not Needed	Relief Not Needed	94,325
Relief Not Found	Relief Not Found	33,955
Overtime	Overtime	52,736
Straight Time	Regular Relief Utilized, Casual at Straight-Time, PT Over FTE, Miscellaneous Straight-Time, PT Employee Moved - Straight-Time, FT Employee Moved - Straight-Time	344,042
Others	Agency, Insufficient Notice, On-Call	18,857

Model Accuracy with "Relief Not Needed"

```
RF.fit(X,y)
print("Random Forest Training Score:", round(RF.score(X,y),3))
print("Random Forest Test Score:", round(RF.score(X_val,y_val),3))
pd.DataFrame([list(RF.feature_importances_)],columns = feature_cols)
```

```
Random Forest Training Score: 0.721
Random Forest Test Score: 0.682
```

5]:

	EXCEPTION_HOURS	NOTICE	JOB_FAMILY	MONTH	WEEKDAY	SITE
0	0.362349	0.396857	0.15944	0.028838	0.015985	0.036532

6]:

```
# create result dataframe
predictions_RF = RF.predict(X_val)
pred_dict = X_val.copy()
pred_dict['EARNING_CATEGORY'] = y_val
pred_dict['RANDOM_FOREST'] = predictions_RF
result = pd.DataFrame(pred_dict)

# display test accuracy for all EARNING_CATEGORY
for i in df["EARNING_CATEGORY"].unique().tolist():
    print("Test accuracy for",i,":",
          round(result[result["EARNING_CATEGORY"]==i][result["RANDOM_FOREST"]==i].shape[0]/
                result[result["EARNING_CATEGORY"]==i].shape[0],3))

result.head(10)
```

```
Test accuracy for Straight Time 0.934
Test accuracy for Relief Not Needed 0.253
Test accuracy for Overtime 0.252
Test accuracy for Relief Not Found 0.224
Test accuracy for Others 0.408
```

Model Accuracy with "Relief Not Needed"

```
RF.fit(X,y)
print("Random Forest Training Score:", round(RF.score(X,y),3))
print("Random Forest Test Score:", round(RF.score(X_val,y_val),3))
pd.DataFrame([list(RF.feature_importances_)],columns = feature_cols)
```

```
Random Forest Training Score: 0.839
Random Forest Test Score: 0.798
```

]:

	EXCEPTION_HOURS	NOTICE	JOB_FAMILY	MONTH	WEEKDAY	SITE
0	0.409057	0.526392	0.009839	0.017791	0.012732	0.024189

]:

```
# create result dataframe
predictions_RF = RF.predict(X_val)
pred_dict = X_val.copy()
pred_dict['EARNING_CATEGORY'] = y_val
pred_dict['RANDOM_FOREST'] = predictions_RF
result = pd.DataFrame(pred_dict)

# display test accuracy for all EARNING_CATEGORY
for i in df["EARNING_CATEGORY"].unique().tolist():
    print("Test accuracy for",i,":",
          round(result[result["EARNING_CATEGORY"]==i][result["RANDOM_FOREST"]==i].shape[0]/
                result[result["EARNING_CATEGORY"]==i].shape[0],3))

result.head(10)
```

```
Test accuracy for Straight Time : 0.967
Test accuracy for Overtime : 0.25
Test accuracy for Relief Not Found : 0.263
Test accuracy for Others : 0.419
```

Challenge and concerns

- Inbalanced data
- Missing vital features
- Unblanced Data wrangling method

Overtime Analysis

- Assume we want to predict number of exceptions backfilled by "overtime" one week ahead
- Use information that is already recorded in the system

Features used:

- Week of year
- Day of week
- Number of exceptions already created

Features to consider:

- Operation hours
- Holiday

Job Family: DC1000

In [3]:

```
# consider only job family DC1
dc1 = raw[(raw['JOB_FAMILY'] == "DC1000")]
dc1.shape
```

Out[3]:

```
(643567, 52)
```

Number of exceptions already created

- `EXCEPTION_CREATION_TO_SHIFTSTART_MINUTES < - 10080`

Weeks and days

- One-hot coding

Linear Regression

Training data:

- 2013, 2015, 2016

Validation data:

- 2017

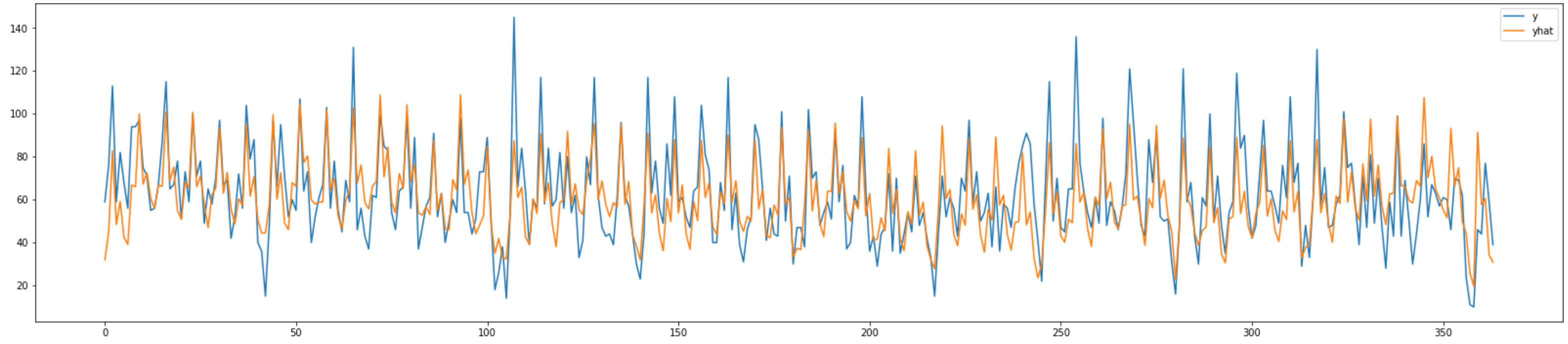
In [9]:
```python
# split train and validation
train = data[(data["year"] < 2017) & (data["year"] != 2014)]
val = data[data["year"] == 2017]

# sizes
print("train :", train.shape[0])
print("val   :", val.shape[0])
```

```
train : 1099
val   : 364
```

In [14]:
```
# check result
result_day['acc'] = np.abs(result_day['y'] - result_day['yhat'])
print("MAE  :",np.mean(result_day['acc']))
```

```
MAE  : 11.260027928138353
```

In [15]:
```
plt.figure(figsize=(28, 6))
# visulize result
plt.plot(result_day.y)
plt.plot(result_day.yhat)
plt.legend()
plt.show()
```



Job Family: DC2A00

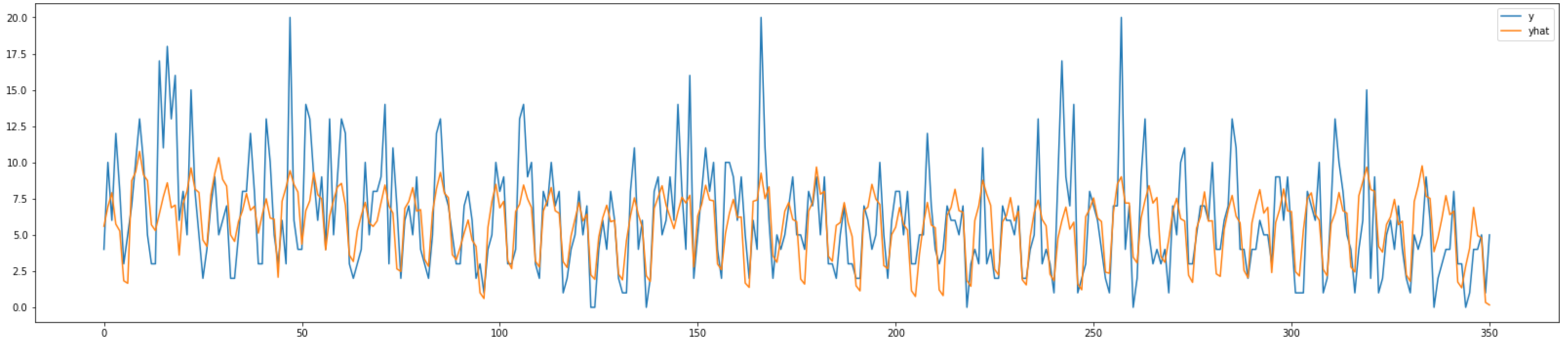
In [20]:

```
# check result
result_day['acc'] = np.abs(result_day['y'] - result_day['yhat'])
print("MAE  :",np.mean(result_day['acc']))
```

```
MAE  : 2.219352526542468
```

In [21]:

```
plt.figure(figsize=(28, 6))
# visulize result
plt.plot(result_day.y)
plt.plot(result_day.yhat)
plt.legend()
plt.show()
```



Job Family: DC2B00

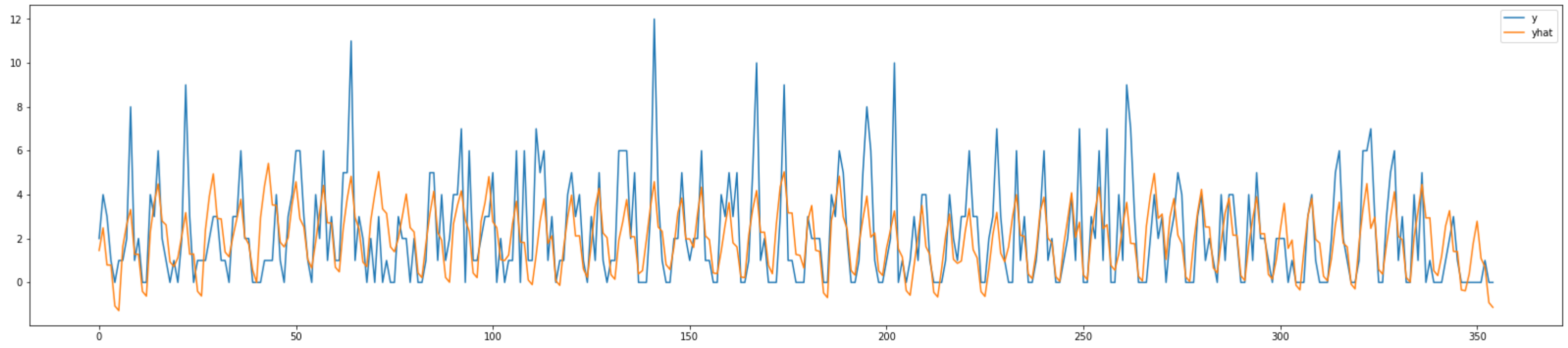
In [24]:

```
# check result
result_day['acc'] = np.abs(result_day['y'] - result_day['yhat'])
print("MAE  :",np.mean(result_day['acc']))
```

```
MAE  : 1.322522832306338
```

In [25]:

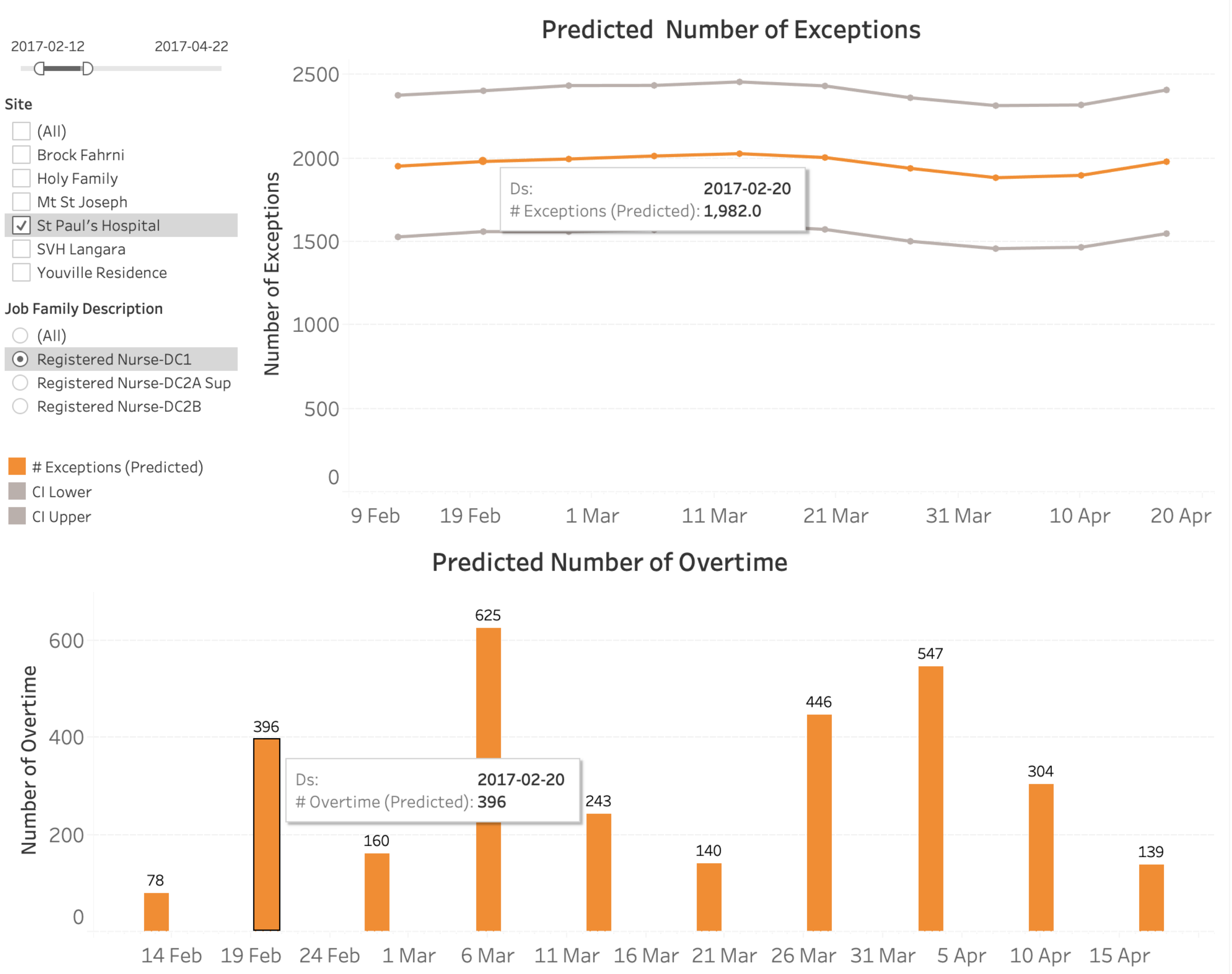
```
plt.figure(figsize=(28, 6))
# visulize result
plt.plot(result_day.y)
plt.plot(result_day.yhat)
plt.legend()
plt.show()
```



Improvements:

- Combine features together
- Add holiday feature
- Combine with other models (random forests), adjust with the predictions

Dashboard Proposal



2017-02-12
2017-04-22
Site
(All)
Brock Fahrni
Holy Family
Mt St Joseph
St Paul's Hospital
SVH Langara
Youville Residence
Job Family Description
(All)
Registered Nurse-DC1
Registered Nurse-DC2A Sup
Registered Nurse-DC2B
Exceptions (Predicted)
CI Lower
CI Upper
Predicted Number of Exceptions
Number of Exceptions
2500
2000
1500
1000
500
0
Ds: 2017-02-20
Exceptions (Predicted): 1,982.0
9 Feb
19 Feb
1 Mar
11 Mar
21 Mar
31 Mar
10 Apr
20 Apr
Predicted Number of Overtime
Number of Overtime
600
400
200
0
78
396
160
625
243
140
446
547
304
139
Ds: 2017-02-20
Overtime (Predicted): 396
14 Feb
19 Feb
24 Feb
1 Mar
6 Mar
11 Mar
16 Mar
21 Mar
26 Mar
31 Mar
5 Apr
10 Apr
15 Apr