専門家知識の利用

18ss319u　オウ　ヨウケイコ

特徴量エンジニアリングでは、特定のアプリケーションに関する専門家知識を利用することができる。多くの場合、機械学習の目的は専門家がルールを設計しなくても済むようにすることだが、だからといって、特定のアプリケーションやドメインに関する事前知識を捨てるべきだということにはならない。ドメインの専門家が、最初に得られるデータ表現よりもはるかに情報量の多い有用な特徴量を特定する手助けをしてくれることは多い。

専門家知識を利用するケースを見ていこう。

ニューヨークにはCiti　Bikeと言うものがあり、登録制のレンタル自転車のステーションのネットワークを運用している。

ここでのタスクは、ある特定の日時に、どのぐらいの人がAndreasの家の前にあるステーションで自転車をレンタルするかを予測する頃だ。これが分かれば、彼が自転車をレンタルできるかどうかを予測できる。

In[49]:

```
citibike = mglearn.datasets.load_citibike()
```

In[50]:

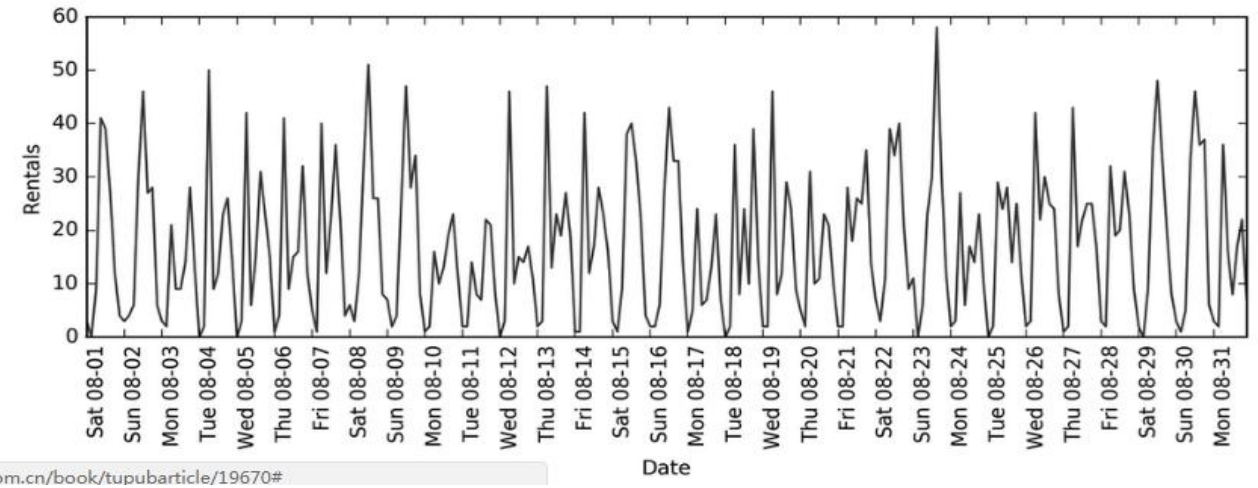
```
print("Citi Bike data:\n{}".format(citibike.head()))
```

Out[50]:

```
Citi Bike data:
starttime
2015-08-01 00:00:00     3.0
2015-08-01 03:00:00     0.0
2015-08-01 06:00:00     9.0
2015-08-01 09:00:00    41.0
2015-08-01 12:00:00    39.0
Freq: 3H, Name: one, dtype: float64
```

In[51]:

```
plt.figure(figsize=(10, 3))
xticks = pd.date_range(start=citibike.index.min(), end=citibike.index.max(),
                       freq='D')
plt.xticks(xticks, xticks.strftime("%a %m-%d"), rotation=90, ha="left")
plt.plot(citibike, linewidth=1)
plt.xlabel("Date")
plt.ylabel("Rentals")
```



まず、2015年8月のこの特定のステーションのデータをpandasのDataFrameとしてロードする。データを3時間ごとの間隔にサンプルし直して、日毎の傾向を見てみよう。

このような時系列に対する予測タスクでは、過去から学習と未来を予測する。つまり、訓練セットとテストセットを分割する際に、ある特定の日までの全てのデータを訓練セットとし、それ以後をテストセットとするのだ。ここでは最初の23日分に相当する184データポイントを訓練セットとし、残りの8日分に相当する64データポイントをテストセットとした。

In[52]:

```
# 提取目标值（租车数量）
y = citibike.values
# 利用"%s"将时间转换为POSIX时间
X = citibike.index.strftime("%s").astype("int").reshape(-1, 1)
```

In[54]:

```
# 使用前184个数据点用于训练，剩余的数据点用于测试
n_train = 184

# 对给定特征集上的回归进行评估和作图的函数
def eval_on_features(features, target, regressor):
    # 将给定特征划分为训练集和测试集
    X_train, X_test = features[:n_train], features[n_train:]
    # 同样划分目标数组
    y_train, y_test = target[:n_train], target[n_train:]
    regressor.fit(X_train, y_train)
    print("Test-set R^2: {:.2f}".format(regressor.score(X_test, y_test)))
    y_pred = regressor.predict(X_test)
    y_pred_train = regressor.predict(X_train)
    plt.figure(figsize=(10, 3))

    plt.xticks(range(0, len(X), 8), xticks.strftime("%a %m-%d"), rotation=90,
               ha="left")

    plt.plot(range(n_train), y_train, label="train")
    plt.plot(range(n_train, len(y_test) + n_train), y_test, '-', label="test")
    plt.plot(range(n_train), y_pred_train, '--', label="prediction train")

    plt.plot(range(n_train, len(y_test) + n_train), y_pred, '--',
             label="prediction test")
    plt.legend(loc=(1.01, 0))
    plt.xlabel("Date")
    plt.ylabel("Rentals")
```

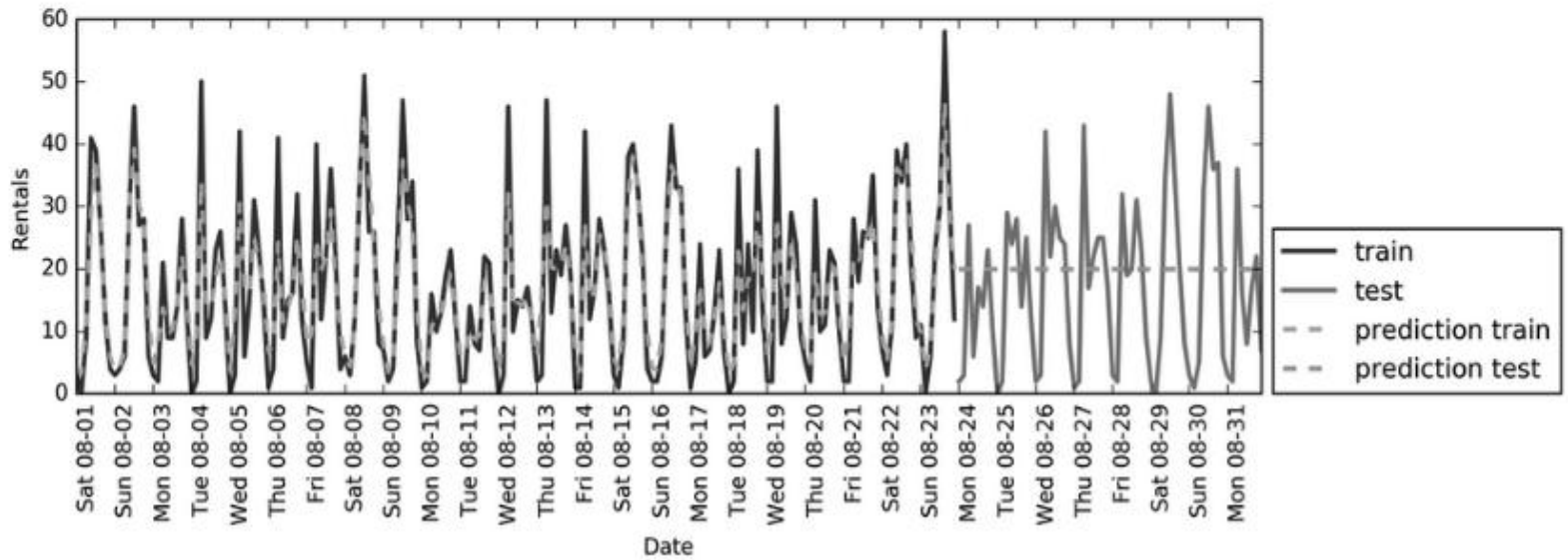
ターゲット値を抽出して、時刻を’%s’でPOSIX時刻に変換する。データを訓練セットとテストセットに分割する関数を定義し、モデルを構築し、結果を可視化する。

In[55]:

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100, random_state=0)
plt.figure()
eval_on_features(X, y, regressor)
```

Out[55]:

```
Test-set R^2: -0.04
```



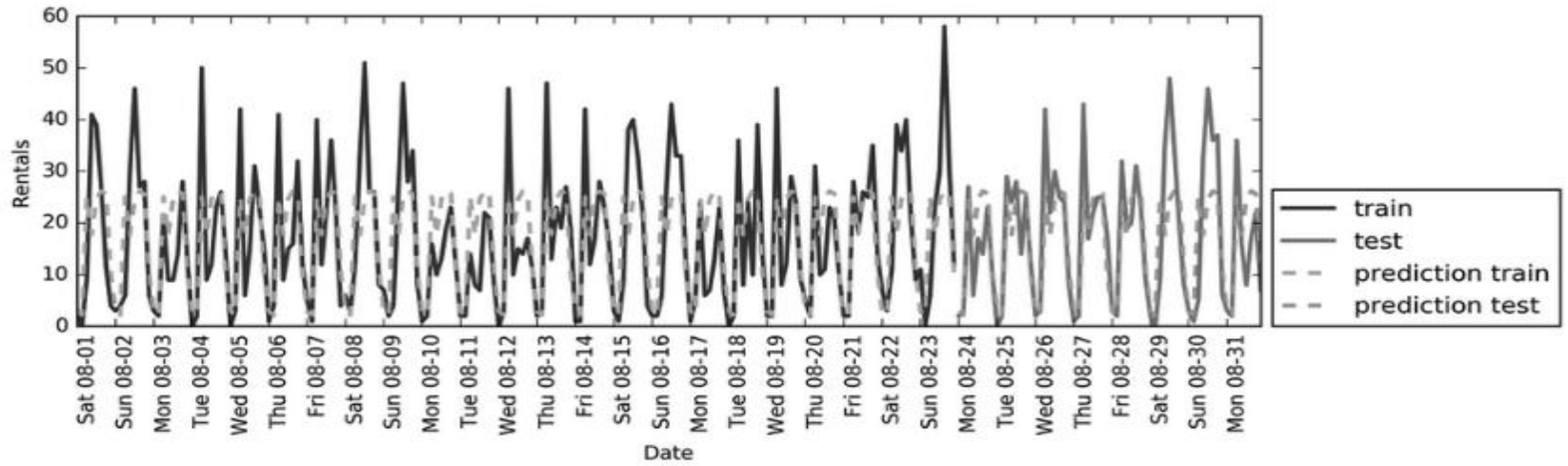
以前見た通り、ランダムフォレストはデータの前処理をほとんど必要としないので、このモデルが最初に扱うモデルとしては良さそうだ。POSIX時間を特徴量Xとし、ランダムフォレスト回帰器とともにeval－on－features関数に与える。

In[56]:

```
X_hour = citibike.index.hour.reshape(-1, 1)
eval_on_features(X_hour, y, regressor)
```

Out[56]:

```
Test-set R^2: 0.60
```



訓練データのレンタル履歴を見ると、二つの要素がとても重要であることがわかる。一日の中の時間帯と曜日である。従って、これらの特徴量を加えてみよう。POSIXを見ても何も分からないのでこの特徴量は落とす。

In[57]:

```
X_hour_week = np.hstack([citibike.index.dayofweek.reshape(-1, 1),
                         citibike.index.hour.reshape(-1, 1)])
eval_on_features(X_hour_week, y, regressor)
```

Out[57]:

```
Test-set R^2: 0.84
```

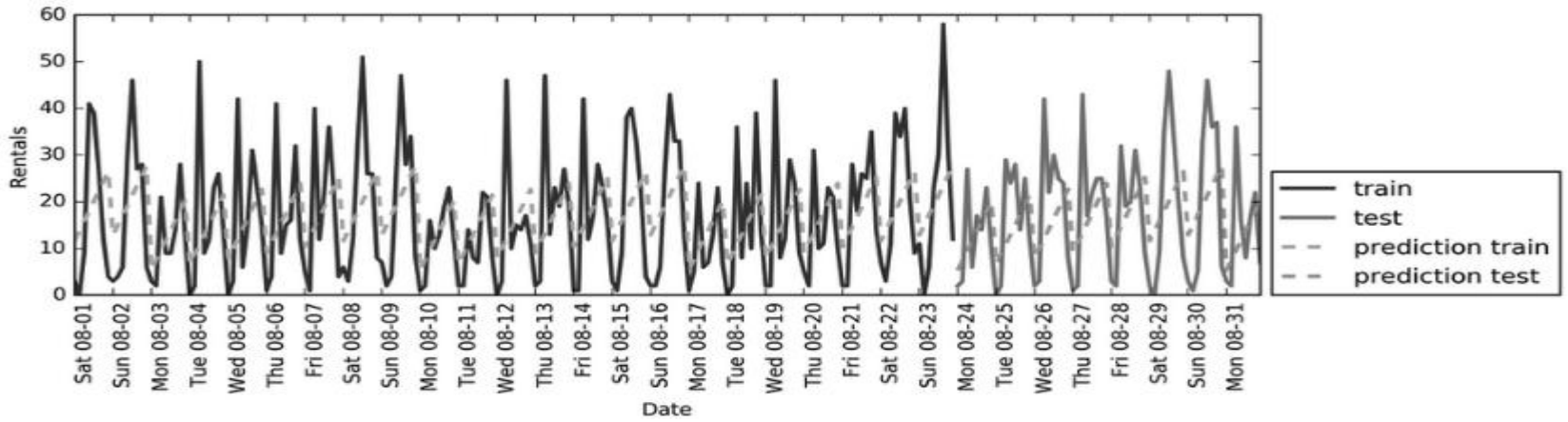
曜日と時刻を考慮に入れた周期的な挙動を捉えたモデルができた。R^2スコアは0．84と予測性能もたかくなっている。
もっと簡単なLinearRegressionで試してみよう。

In[58]:

```
from sklearn.linear_model import LinearRegression
eval_on_features(X_hour_week, y, LinearRegression())
```

Out[58]:

```
Test-set R^2: 0.13
```



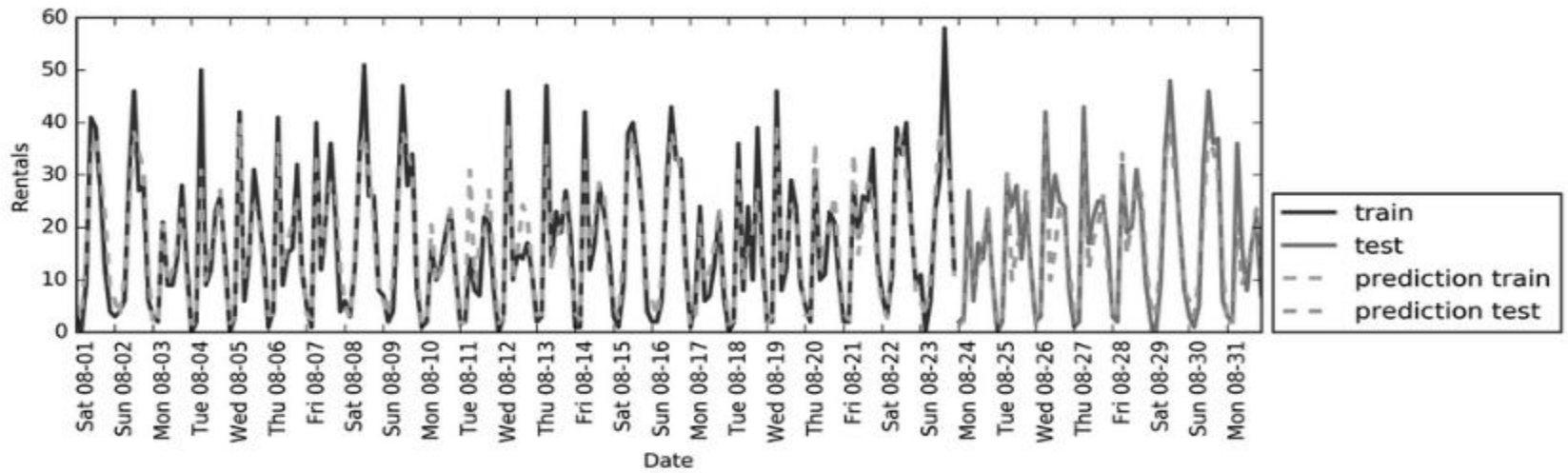
LinearRegressionの性能はずっと悪いし、周期パターンも妙だ。これは曜日や時刻が整数でエンコードされていて、連続値として解釈されているからだ。整数をOneHotEncoderを用いて変換することで、カテゴリ変数として解釈すれば、パターンを捉えることができる。

In[61]:

```
poly_transformer = PolynomialFeatures(degree=2, interaction_only=True,
                                      include_bias=False)
X_hour_week_onehot_poly = poly_transformer.fit_transform(X_hour_week_onehot)
lr = Ridge()
eval_on_features(X_hour_week_onehot_poly, y, lr)
```

Out[61]:

```
Test-set R^2: 0.85
```



ここで交互作用特徴量を用いれば、曜日と時刻の組み合わせに対して係数を学習させることができる。

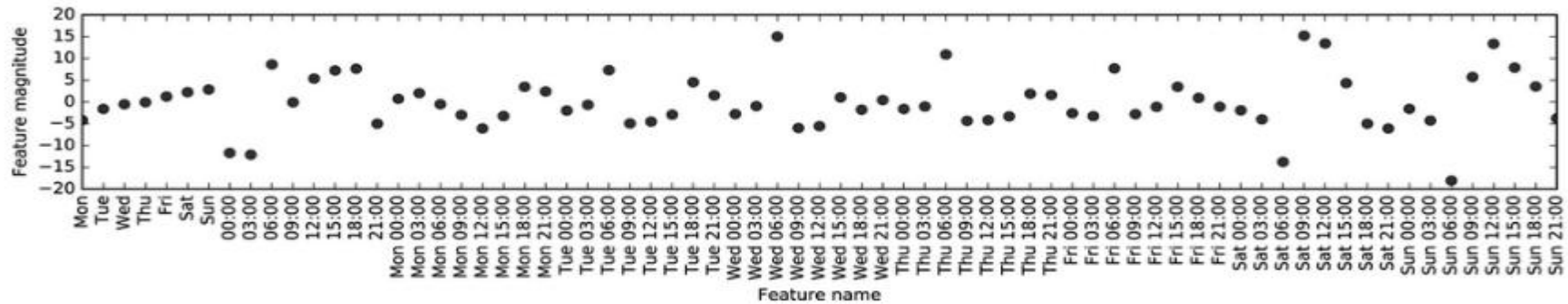
In[63]:

```
features_poly = poly_transformer.get_feature_names(features)
features_nonzero = np.array(features_poly)[lr.coef_ != 0]
coef_nonzero = lr.coef_[lr.coef_ != 0]
```

下面将线性模型学到的系数可视化，如图 4-19 所示：

In[64]:

```
plt.figure(figsize=(15, 2))
plt.plot(coef_nonzero, 'o')
plt.xticks(np.arange(len(coef_nonzero)), features_nonzero, rotation=90)
plt.xlabel("Feature name")
plt.ylabel("Feature magnitude")
```



この変換によって、ようやくランダムフォレストと同等の性能になった。線形モデルで学習された係数を可視化したものを図に示す。