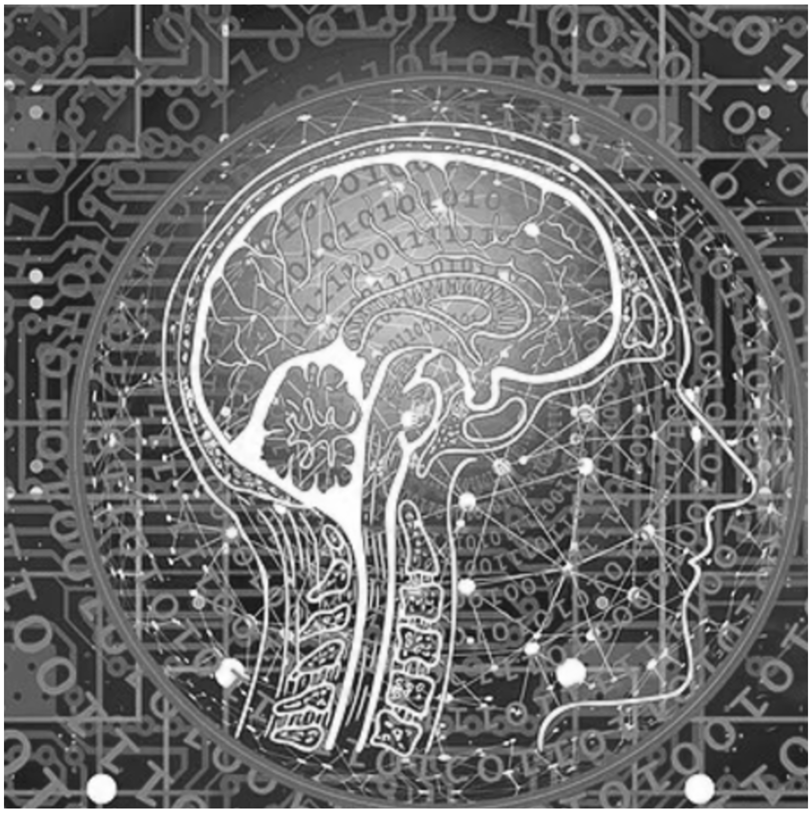


Coursera IBM-- Supervised Machine Learning: Regression Final Project

By: Liam Webster

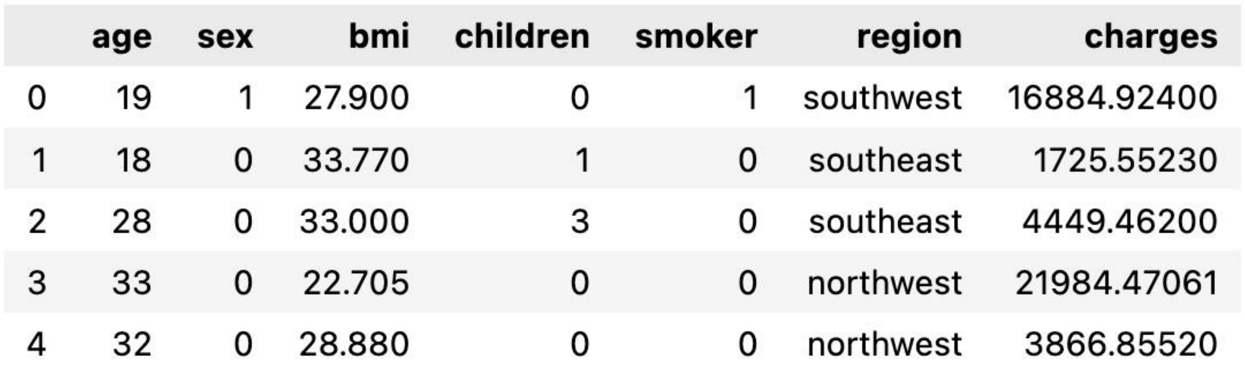
6/13/2022

Objective:

The main objective of this analysis was to explore the predictability of individual medical expenditure via a select few parameters. Specifically factors such as age, bmi, smoking history, and number of kids was taken into account. This model could then be used both in predictability or interpretation applications. Insurance companies would be interested in the predictability applications so they could accurately quote new customers. Consumers would be interested in the interpretation applications so they could address their attributes/risks and lower their expected medical expenditure.

Data:

The dataset chosen comes from ACME Insurance Inc. The dataset has 1338 rows/observations and 7 columns/features-- BMI, number of children, smoking history, region, and individual expenditure.



	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.92400
1	18	0	33.770	1	0	southeast	1725.55230
2	28	0	33.000	3	0	southeast	4449.46200
3	33	0	22.705	0	0	northwest	21984.47061
4	32	0	28.880	0	0	northwest	3866.85520

…

HARSH SINGH. (2022, March). Medical Insurance Payout, 1.0. Retrieved June 12th 2022 from https://www.kaggle.com/datasets/harshsingh2209/medical-insurance-payout.

Approach:

Firs a general inspection of the data set was performed-- via both plots and manual inspection. The couple rows containing NaN data were dropped. The ‘sex’ and ‘smoker’ column were manually encoded-- 1 for female/0 for male and 1 for smoker/0 for non-smoker, respectively. A copy of the data frame was made and with said copy the ‘region’ column was one-hot encoded; with the other copy the ‘region’ column was dropped. Thus resulting into two clean data frames one with the original columns except ‘region’ referred to as data frame 1, and one with the original columns except ‘region’ is one-hot encoded referred to as data frame 2.

Models:

First training and test splits were created with a 70 : 30 ratio respectively. These were declared globally and were used in each of the regression models. The first model trained was a simple linear regression model. Data frame 1 resulted in a 0.767 R2 score while data frame 2 resulted in a 0.771 R2 score. As this difference is quite negligible in comparison to the complexity introduced by one-hot encoding the ‘region‘ column further testing on data frame 2 was halted. The next model trained was a linear regression model with standard scaling. This model was less accurate and produced an R2 score of 0.694. This is likely due to already low variance in the observations. The next model trained was a linear regression model with polynomial features. A degree 20 polynomial feature resulted in a negative R2 score, but a degree 2 polynomial feature resulted in an R2 score of 0.845. The next model trained was a Ridge and Lasso regularization model which after tuning both resulted in an R2 score of 0.845. The last model trained was an ElasticNet model that resulted in an R2 score of 0.769.

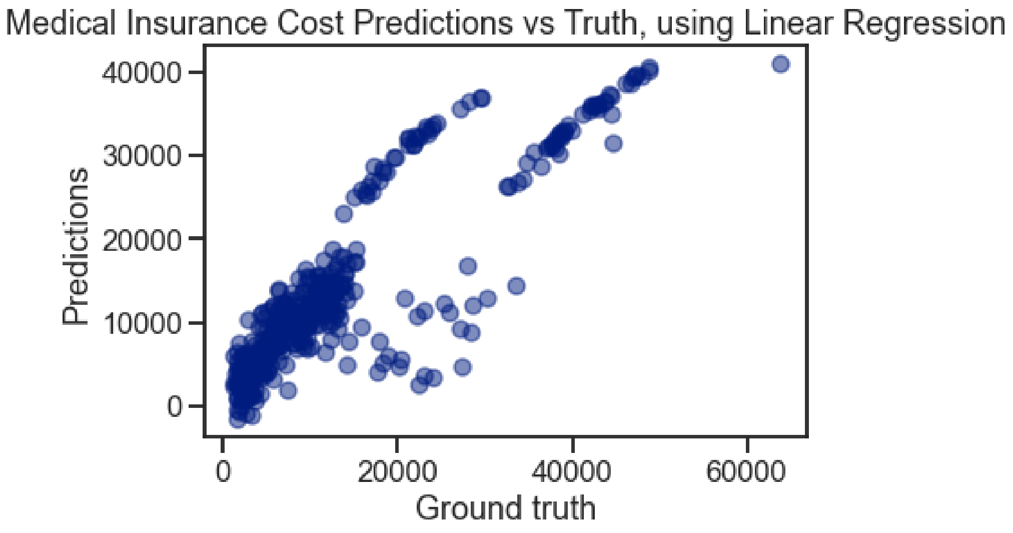
Key Findings:

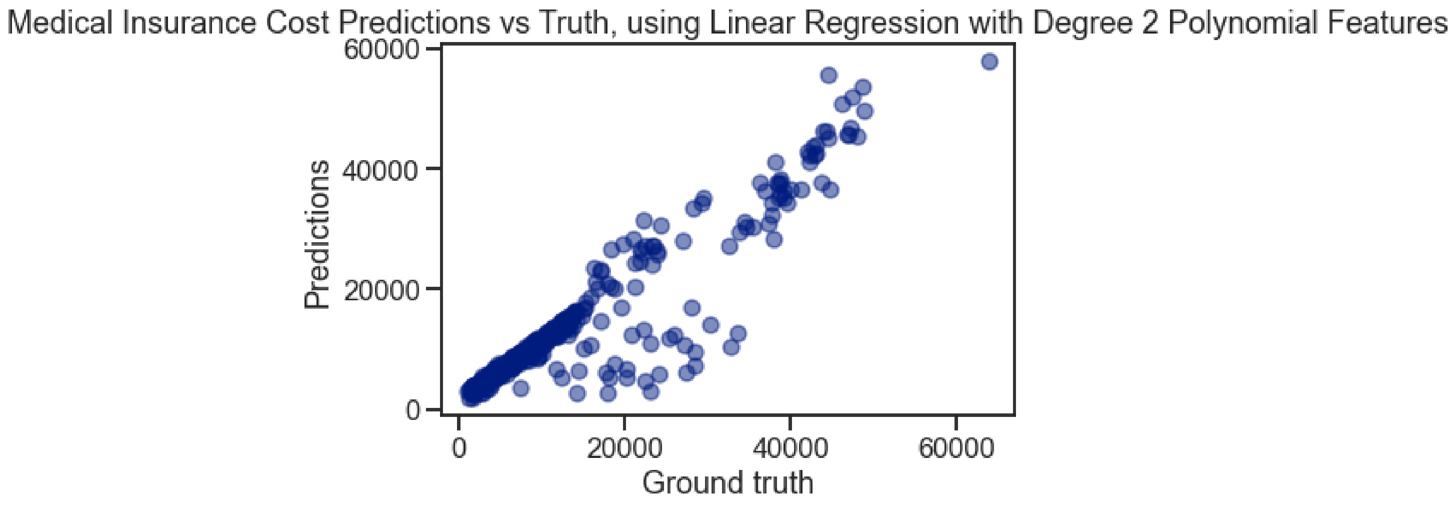
The best model-- minimizing variance and bias --was the linear regression model with degree 2 polynomial featuring. It resulted in the highest R2 score while also relaying good interpretability. As can be seen in the graphs in Appendix A; the polynomial transformation helped to normalize the right skewed data.
The degree 2 polynomial featuring accounted for a diminishing return trend that the linear model couldn’t account for. There was a flattening out in medical expenditure even as the risk factors continued to increase. Each of the 6 factors contributed in similar magnitude to the model. Thus Lasso or Ridge was not able to zero any of the coefficients without underfitting the model. Thus it can be concluded that the base linear regression model was not complex enough and underfit the data and a degree 20 polynomial feature introduced too much complexity and overfit the data. A degree 2 polynomial feature introduced the optimum amount of complexity, positioning the model in the minimization point of bias and variance error.

Further Research:

To further this research the next steps to take would be to further validate the findings. Possibly using k-fold cross validation on our degree 2 polynomial linear regression model. Also trying different scaling methods such as log scaling could present new findings. Another step to take would be to reintroduce the one-hot encoded data frame into our degree 2 polynomial linear regression model.

Appendix A:


Medical Insurance Cost Predictions vs Truth, using Linear Regression
Predictions
40000
30000
20000
10000
0
0
20000
40000
60000
Ground truth


Medical Insurance Cost Predictions vs Truth, using Linear Regression with Degree 2 Polynomial Features
Predictions
60000
40000
20000
0
0
20000
40000
60000
Ground truth

Appendix B:

Untitled

June 13, 2022

[ ]:
```python
# Imports

import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNetCV, LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
```

[ ]:
```python
# Data Read

warnings.simplefilter("ignore")

filepath = 'Data/insurance.csv'
data = pd.read_csv(filepath)
```

[ ]:
```python
# Data Cleaning

data_clean = data.copy()
data_clean.dropna(axis=0,inplace=True)

for i in range(len(data_clean)):
    if data_clean['sex'][i] == 'female':
        data_clean['sex'][i] = 1
    elif data_clean['sex'][i] == 'male':
        data_clean['sex'][i] = 0
    else:
        data_clean['sex'][i] = 3
    if data_clean['smoker'][i] == 'yes':
        data_clean['smoker'][i] = 1
    elif data_clean['smoker'][i] == 'no':
        data_clean['smoker'][i] = 0
    else:
        data_clean['smoker'][i] = 3
```

```
data_clean['sex'] = pd.to_numeric(data_clean['sex'])
data_clean['smoker'] = pd.to_numeric(data_clean['smoker'])

data_clean_ohc_final = pd.get_dummies(data_clean, columns=['region'])
data_clean_ohc_final.columns = data_clean_ohc_final.columns.str.
 ↪replace('region_', '')
data_clean_final = data_clean.drop(['region'], axis=1)
```

```
[ ]: # Create Train and Test Splits

y_col = 'charges'

feature_cols = [x for x in data_clean_final if x != y_col]
X_data = data_clean_final[feature_cols]
y_data = data_clean_final[y_col]

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
                                                    test_size=0.3,␣
 ↪random_state=42)
feature_cols = [x for x in data_clean_ohc_final.columns if x != y_col]
X_data_ohc = data_clean_ohc_final[feature_cols]
y_data_ohc = data_clean_ohc_final[y_col]

X_train_ohc, X_test_ohc, y_train_ohc, y_test_ohc = train_test_split(X_data_ohc,␣
 ↪y_data_ohc,
                                                    test_size=0.3,␣
 ↪random_state=42)
```

```
[ ]: # Linear Regression Model with no Scaling

LR = LinearRegression()

error_df = list()

LR = LR.fit(X_train, y_train)
y_train_pred = LR.predict(X_train)
y_test_pred = LR.predict(X_test)

error_df.append(pd.Series({'train': r2_score(y_train, y_train_pred),
                           'test' : r2_score(y_test,  y_test_pred)},
                           name='no enc'))

LR = LR.fit(X_train_ohc, y_train_ohc)
y_train_ohc_pred = LR.predict(X_train_ohc)
y_test_ohc_pred = LR.predict(X_test_ohc)

error_df.append(pd.Series({'train': r2_score(y_train_ohc, y_train_ohc_pred),
                            'test' : r2_score(y_test_ohc,  y_test_ohc_pred)},
                           name='one-hot enc'))

# Assemble the results
error_df = pd.concat(error_df, axis=1)
error_df
```

[ ]:
```
# Plot Results-- Linear Regression(No Scaling)

sns.set_context('talk')
sns.set_style('ticks')
sns.set_palette('dark')

ax = plt.axes()
# we are going to use y_test, y_test_pred
ax.scatter(y_test, y_test_pred, alpha=.5)

ax.set(xlabel='Ground truth',
       ylabel='Predictions',
       title='Medical Insurance Cost Predictions vs Truth, using Linear␣
 ↪Regression');
```

[ ]:
```
# Linear Regression Model with Standard Scaling

s = StandardScaler()
LR_s = LinearRegression()

X_train_s = s.fit_transform(X_train)
X_test_s = s.transform(X_test)
LR_s.fit(X_train_s, y_train)
y_train_s = LR_s.predict(X_train_s)
y_pred_s = LR_s.predict(X_test_s)


error_df = list()

error_df.append(pd.Series({'train': r2_score(y_train_s, y_train),
                            'test' : r2_score(y_pred_s, y_test)},
                           name='no enc'))
error_df = pd.concat(error_df, axis=1)
error_df
```

[ ]:
```
# Plot Results-- Linear Regression with Standard Scaling

sns.set_context('talk')
sns.set_style('ticks')
sns.set_palette('dark')
```

```
ax = plt.axes()
# we are going to use y_test, y_test_pred
ax.scatter(y_test, y_pred_s, alpha=.5)

ax.set(xlabel='Ground truth',
       ylabel='Predictions',
       title='Medical Insurance Cost Predictions vs Truth, using Linear␣
 ↪Regression with Standard Scaling');
```

[ ]:
```
# Linear Regression with Polynomial Features

degree = 2
pf = PolynomialFeatures(degree)
lr = LinearRegression()

X_train_poly = pf.fit_transform(X_train_s)
X_test_poly = pf.transform(X_test_s)
lr = lr.fit(X_train_poly, y_train)
y_pred_poly = lr.predict(X_test_poly)
r2_score(y_pred_poly, y_test)
print(lr.coef_)

ax = plt.axes()
ax.scatter(y_test,y_pred_poly, alpha=0.5)
ax.set(xlabel='Ground truth',
       ylabel='Predictions',
       title='Medical Insurance Cost Predictions vs Truth, using Linear␣
 ↪Regression with Degree 2 Polynomial Features');
```

[ ]:
```
# %% Linear Regression with Ridge and Lasso Regularization

rr = Ridge(alpha=0.00000000000001)
rr = rr.fit(X_train_poly, y_train)
y_pred_rr = rr.predict(X_test_poly)
print(r2_score(y_pred_rr, y_test))
print(rr.coef_)

# The lasso regression model
lassor = Lasso(alpha=0.0000000000001)
lassor = lassor.fit(X_train_poly, y_train)
y_pred_lr = lassor.predict(X_test_poly)
print(r2_score(y_pred_lr, y_test))
print(lassor.coef_)
```

[ ]:
```
# ElasticNet Model
l1_ratios = np.linspace(0.01, 0.4, 20)
```

```python
alphas2 = np.array([1e-5, 5e-5, 0.0001, 0.0005, 1])

elasticNetCV = ElasticNetCV(alphas=alphas2,
                            l1_ratio=l1_ratios,
                            max_iter=1e4).fit(X_train_ohc, y_train_ohc)
elasticNetCV_rmse = r2_score(y_test_ohc, elasticNetCV.predict(X_test_ohc))

print(elasticNetCV.alpha_, elasticNetCV.l1_ratio_, elasticNetCV_rmse)
```