Untitled

June 13, 2022

[ ]:
```
# 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
```

[ ]:
```
# Data Read

warnings.simplefilter("ignore")

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

[ ]:
```
# 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)
```

```
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)
```