Assignment 5

1. Choose a regression dataset (bikeshare is allowed), perform a test/train split, and build a regression model (just like in assingnment 3), and calculate the
   - Training Error (MSE, MAE)
   - Testing Error (MSE, MAE)
2. Choose a classification dataset (not the adult.data set, The UCI repository has many datasets as well as Kaggle), perform test/train split and create a classification model (your choice but DecisionTree is fine). Calculate
   - Accuracy
   - Confusion Matrix
   - Classifcation Report
3. (Bonus) See if you can improve the classification model's performance with any tricks you can think of (modify features, remove features, polynomial features)

1) Regression model on insurance data

Data from GitHub for book Machine Learning with R by Brett Lantz.
[https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv (https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv)](https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv)

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [2]:
```
insurance = pd.read_csv('../data/insurance.csv')
insurance.head()
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [3]:
```
insurance.columns
```

Out[3]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype
='object')
```

In [4]:
```
x = insurance[['bmi']]
y = insurance[['charges']]
```

In [5]:
```
linear = linear_model.LinearRegression()
linear.fit(x, y)
(linear.coef_, linear.intercept_)
```

Out[5]:
```
(array([[393.8730308]]), array([1192.93720896]))
```

In [6]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25)
```

In [7]:
```
x.shape, x_train.shape, x_test.shape
```

Out[7]:
```
((1338, 1), (1003, 1), (335, 1))
```

In [8]:
```
model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_
```

Out[8]:
```
(array([[371.44701056]]), array([1929.30616021]))
```

In [9]:
```
mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept_)
```

Out[9]:
```
136135713.62918454
```

In [10]:
```
mean_absolute_error(y_test, np.dot(x_test, model.coef_) + model.intercept_)
```

Out[10]:
```
8981.315252084027
```

2) Classification on insurance dataset

In [11]:
```
# checking for unique characters
insurance['smoker'].unique()
```

Out[11]:
```
array(['yes', 'no'], dtype=object)
```

In [12]:
```
non_numeric_columns = ['sex', 'region']
x = insurance.copy().drop(non_numeric_columns, axis=1)
x
```

Out[12]:

	age	bmi	children	smoker	charges
0	19	27.900	0	yes	16884.92400
1	18	33.770	1	no	1725.55230
2	28	33.000	3	no	4449.46200
3	33	22.705	0	no	21984.47061
4	32	28.880	0	no	3866.85520
...	...	...	...	...	...
1333	50	30.970	3	no	10600.54830
1334	18	31.920	0	no	2205.98080
1335	18	36.850	0	no	1629.83350
1336	21	25.800	0	no	2007.94500
1337	61	29.070	0	yes	29141.36030

1338 rows × 5 columns

In [13]:
```
x['smoker'] = x.smoker.str.contains('yes').astype(int)
```

In [14]:
```
x.smoker.value_counts()
```

Out[14]:
```
0    1064
1     274
Name: smoker, dtype: int64
```

In [15]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [16]:
```
model = DecisionTreeClassifier(criterion='entropy')
model.fit(x.drop(['smoker'], axis=1), x.smoker)
```

Out[16]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [17]:
```
list(zip(x.drop(['smoker'], axis=1).columns, model.feature_importances_))
```

Out[17]:
```
[('age', 0.05034195160627059),
 ('bmi', 0.13222783770450766),
 ('children', 0.011107480153152032),
 ('charges', 0.8063227305360697)]
```

Using test_train_split

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(x.drop(['smoker'], axis=1),x.:
```

In [19]:
```
x.shape, x_train.shape, x_test.shape
```

Out[19]:
```
((1338, 5), (669, 4), (669, 4))
```

In [20]:
```
model.fit(x_train, y_train)
```

Out[20]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [21]:
```
test_predictions = model.predict(x_test)
```

Calculating accuracy.

In [22]:
```
from sklearn.metrics import (accuracy_score,
                             classification_report,
                             confusion_matrix, auc, roc_curve
                            )
```

In [23]:
```
accuracy_score(y_test, test_predictions)
```

Out[23]:
```
0.9506726457399103
```

Confusion matrix

In [24]:
```
confusion_matrix(y_test, test_predictions)
```

Out[24]:
```
array([[507,  18],
       [ 15, 129]], dtype=int64)
```

Classification report.

In [25]:
```
print(classification_report(y_test, test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       525
           1       0.88      0.90      0.89       144

    accuracy                           0.95       669
   macro avg       0.92      0.93      0.93       669
weighted avg       0.95      0.95      0.95       669
```

3) Removed additional feature (children) slightly improved model

In [26]:
```
removed_features = ['sex', 'children', 'region']
x2 = insurance.copy().drop(removed_features, axis=1)
x2
```

Out[26]:

	age	bmi	smoker	charges
0	19	27.900	yes	16884.92400
1	18	33.770	no	1725.55230
2	28	33.000	no	4449.46200
3	33	22.705	no	21984.47061
4	32	28.880	no	3866.85520
...	...	...	...	...
1333	50	30.970	no	10600.54830
1334	18	31.920	no	2205.98080
1335	18	36.850	no	1629.83350
1336	21	25.800	no	2007.94500
1337	61	29.070	yes	29141.36030

1338 rows × 4 columns

In [27]:
```
x2['smoker'] = x2.smoker.str.contains('yes').astype(int)
x2.smoker.value_counts()
```

Out[27]:
```
0    1064
1     274
Name: smoker, dtype: int64
```

In [28]:
```
model = DecisionTreeClassifier(criterion='entropy')
model.fit(x2.drop(['smoker'], axis=1), x2.smoker)
```

Out[28]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [29]:
```
list(zip(x2.drop(['smoker'], axis=1).columns, model.feature_importances_))
```

Out[29]:
```
[('age', 0.05490254767566018),
 ('bmi', 0.13254866487144468),
 ('charges', 0.8125487874528952)]
```

In [30]:
```
x_train, x_test, y_train, y_test = train_test_split(x2.drop(['smoker'], axis=1),x
```

In [31]:
```
x.shape, x_train.shape, x_test.shape
```

Out[31]:
```
((1338, 5), (669, 3), (669, 3))
```

In [32]:
```
model.fit(x_train, y_train)
```

Out[32]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [33]:
```
test_predictions = model.predict(x_test)
```

In [34]:
```
accuracy_score(y_test, test_predictions)
```

Out[34]:
```
0.9596412556053812
```

In [35]:
```
confusion_matrix(y_test, test_predictions)
```

Out[35]:
```
array([[510,  18],
       [  9, 132]], dtype=int64)
```

In [36]:
```
print(classification_report(y_test, test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       528
           1       0.88      0.94      0.91       141

    accuracy                           0.96       669
   macro avg       0.93      0.95      0.94       669
weighted avg       0.96      0.96      0.96       669
```