US Health Insurance Machine Learning

By Joshua Nguyen

This uses the US Health Insurance dataset which can be downloaded at https://www.kaggle.com/teertha/ushealthinsurancedataset. This mimics with the project done for the dataset downloaded at https://www.kaggle.com/smogomes/prostate-cancer-prediction-model

Import Libraries

In [93]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import sklearn
from  sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold, cross_val_score
import os
```

Import Dataset

In [94]:

```
my_data = pd.read_csv("C:/Users/Josh/Documents/Data Analyst/Rlife/insurance.csv")
my_data1 = pd.read_csv("C:/Users/Josh/Documents/Data Analyst/Rlife/insurance.csv")
```

In [95]:

```
my_data.head() # getting first four rows
```

Out[95]:

	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 [96]:

```
my_data.info() # no null values in our dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
```

```
 ---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In [97]:

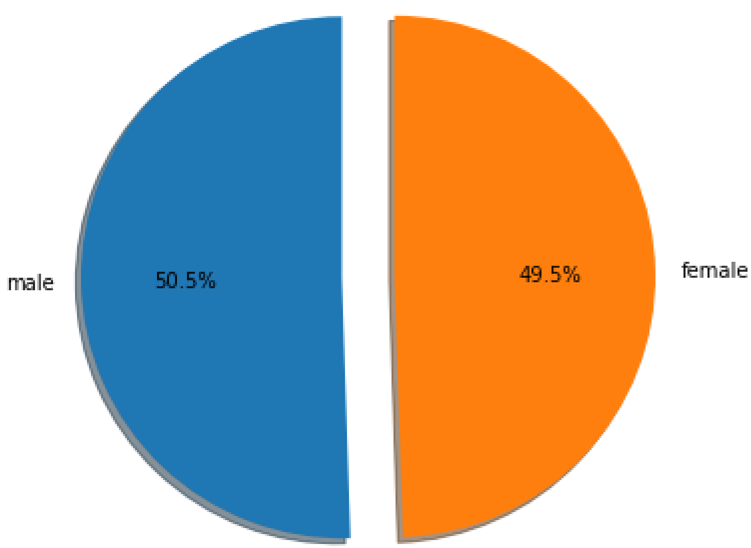
```
my_data.describe() # Five Number Summary
```

Out[97]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [98]:

```
data_dr = my_data['sex'].value_counts()
label = [data_dr.index.tolist()]
plt.pie(data_dr, labels=label[0], shadow=True, explode=(0.0, 0.2), autopct='%1.1f%%', s
plt.gcf().set_size_inches(12,6)
plt.show()
```



The distribution of males and females in our dataset are relatively similar.

In [99]:

```
# Changing sex and smoking to dichtomous numerical variable - dummy variable

#my_data2 = my_data['sex'].replace({'male':0, 'female':1}, inplace=True)

mapping = {'male': 0, 'female':1}

my_data2 = my_data1.replace({'sex': mapping})

mapping2 = {'no': 0, 'yes': 1 }

my_data3 = my_data2.replace({'smoker': mapping2})

my_data3.head()


# male - 0
# female - 1
# non-smoker - 0
# smoker - 1
```

Out[99]:

	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

In [100…

```
# Data Correlation Matrix

corr_matrix = my_data3.corr()
corr_matrix.style.background_gradient()
```

Out[100…

	age	sex	bmi	children	smoker	charges
age	1.000000	0.020856	0.109272	0.042469	-0.025019	0.299008
sex	0.020856	1.000000	-0.046371	-0.017163	-0.076185	-0.057292
bmi	0.109272	-0.046371	1.000000	0.012759	0.003750	0.198341
children	0.042469	-0.017163	0.012759	1.000000	0.007673	0.067998
smoker	-0.025019	-0.076185	0.003750	0.007673	1.000000	0.787251
charges	0.299008	-0.057292	0.198341	0.067998	0.787251	1.000000

Split and Train Sets

In [101…

```
# Can drop region column as it does not provide any information to our analysis

my_data3 = my_data3.drop(['region'], axis=1)
```

In [102…

```
# Split the Data in Train and Test Function

X = my_data3.drop(['smoker'], axis=1) # features
y = my_data3['smoker'] # labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
```

Machine Learning Models

In [103…

```
# Logistic Regressional Model

logreg = LogisticRegression()
logreg.fit(X_train,y_train)
pred_log = logreg.predict(X_test)
```

In [104…

```
# Random Forest Classifier

forest = RandomForestClassifier(n_estimators=50)
forest.fit(X_train, y_train)
pred_forest = forest.predict(X_test)
```

In [105…

```
# K-Neighborhoods Classifier

kn = KNeighborsClassifier(n_neighbors=2)
kn.fit(X_train,y_train)
pred_kn = kn.predict(X_test)
```

In [106…

```
# Decision Tree Classifier

tree = DecisionTreeClassifier()
tree.fit(X_train,y_train)
pred_tree = tree.predict(X_test)
```

In [107…

```
# Report

class_rep_log = classification_report(y_test, pred_log)
class_rep_forest = classification_report(y_test, pred_forest)
class_rep_kn = classification_report(y_test, pred_kn)
class_rep_tree = classification_report(y_test, pred_tree)

# Print Report

print("Logistic Regression: \n", class_rep_log)
print("Forest Classifier: \n", class_rep_forest)
print("KNeighbors Classifier: \n", class_rep_kn)
print("Decision Tree: \n", class_rep_tree)
```

```
Logistic Regression:
               precision    recall  f1-score   support

           0       0.95      0.95      0.95       211
```

```
           1       0.82      0.81      0.81        57

    accuracy                           0.92       268
   macro avg       0.88      0.88      0.88       268
weighted avg       0.92      0.92      0.92       268

Forest Classifier:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       211
           1       0.86      0.98      0.92        57

    accuracy                           0.96       268
   macro avg       0.93      0.97      0.95       268
weighted avg       0.97      0.96      0.96       268

KNeighbors Classifier:
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       211
           1       0.76      0.65      0.70        57

    accuracy                           0.88       268
   macro avg       0.83      0.80      0.81       268
weighted avg       0.88      0.88      0.88       268

Decision Tree:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       211
           1       0.95      0.91      0.93        57

    accuracy                           0.97       268
   macro avg       0.96      0.95      0.95       268
weighted avg       0.97      0.97      0.97       268
```

In [108…

```
kf = KFold(10)

logit_score = cross_val_score(logreg, X, y, cv=kf)
forest_score = cross_val_score(forest, X, y, cv=kf)
KNeighbors_score = cross_val_score(kn, X, y, cv=kf)
tree_score = cross_val_score(tree, X, y, cv=kf)

# Print the mean of each array of scores
print("Logistic Regression:", np.mean(logit_score), '\n'
      "Forest Classification:", np.mean(forest_score), '\n'
      "KNeighbors Classification:", np.mean(KNeighbors_score), '\n'
      "Decision Tree:", np.mean(tree_score), '\n'
     )
```

```
Logistic Regression: 0.9312703400291775
Forest Classification: 0.9649029289642016
KNeighbors Classification: 0.8923745931994164
Decision Tree: 0.9603748176411177
```

The highest score is obtained via the Forest

Classification followed by the Decision Tree, then Logistic Regression and finially KNeighbors Classification