I. SOFTWARE PACKAGE DESCRIPTION

1. PREDICTION

1.1 Data Preprocessing

In order to help patients to better choose a hospital, we do prediction about the total cost on two types of hospitals: specialty hospitals and general hospitals, and two types of surgeries: knee and hip. We use four machine learning algorithms to do the prediction, splitting the data into test and validation sets.

We use a function score_to_numeric to change all categorical data to numeric data while reading the data. Also, we change all columns to integer so that we can do regression and classification to the dataset.

```
row[gender] = score_to_numeric(row[gender],0)
row[race] = score_to_numeric(row[race],0)
row[admi] = score_to_numeric(row[admi],0)
row[disp] = score_to_numeric(row[disp],0)
row[age] = score_to_numeric(row[age],1)
row[price] = score_to_numeric(row[price],2)
for i in range(0, len(row)):
    row[i] = float(row[i])
```

Since the data we use is raw data, we choose two types of feature selection method considering the sparsity of the dataset. Besides, we tested the difference between using Principal Component Analysis (PCA) and without using PCA on the rows.

1.1.1. Variance Feature Selection

VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

```
def varianceselect(x):
    sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
    X_new = sel.fit_transform(x)
    print X_new.shape,"xnewshape"
    return X_new
```

1.1.2. Tree Selection

Tree-based estimators can be used to compute feature importances, which in turn can be used to discard irrelevant features.

```
def treeselect(x):
    clf_tree = ExtraTreesClassifier()
    clf_tree = clf_tree.fit(x, y)
    print clf_tree.feature_importances_
    model_tree = SelectFromModel(clf_tree, prefit=True)
    X_new = model_tree.transform(x)
    print X_new.shape,"xnewshape"
    return X_new, clf_tree.feature_importances_
```

1.1.3. L1-based feature selection

Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with feature_selection.SelectFromModel to select the non-zero coefficients.

```
def L1select(x):
    lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(x, y)
    model = SelectFromModel(lsvc, prefit=True)
    X_new = model.transform(x)
    print X_new.shape,"xnewshape"
    return X_new
```

1.1.4. PCA

Principal component analysis (PCA) is a method of linear dimensionality reduction using Singular Value Decomposition of the data and keeping only the most significant singular vectors to project the data to a lower dimensional space. Here we use PCA on rows instead of columns to avoid data sparsity, that is, to filter patients who have chosen a hospital that is deviated a lot from choices of most people.

```
#use PCA to do dimension reduction
def dimensionreducPCA(data):
    data_mn = np.mean(data, 0)
    data = data - np.repeat(data_mn.reshape(1, -1), data.shape[0], 0)

    B = np.dot(data.T, data)
    [val,vec] = np.linalg.eigh(B)
    D = vec[:, np.argsort(val)[::-1]]
    c = np.dot(D.T, data.T)
    c = c[:,:1000]

    data = np.dot(c.T, D.T)
    data = data + np.repeat(data_mn.reshape(1, -1), data.shape[0], 0)
    return data
```

Considering the run time of selection, we choose two types of selection: variance selection and tree selection to do validation.

1.2 Algorithms

1.2.1. Linear regression

For linear regression, we used cost as Y and all the other columns as X ($x_1, x_2, \ldots, x_9$). The goal is to predict cost Y from X using a linear function.

We use least square (LS) estimator to calculate the parameters β and we have

$$\beta = (X^T X)^{-1} X^T Y$$

which can minimize the least square error.

```
#linear regression
LR = linear_model.LinearRegression()
LR.fit (training[:,0:l], training[:,l])
res1 = LR.predict(testing[:,0:l])
```

1.2.2. Ridge regression

When we have too many predictors, it may cause linear least squares estimator to have low bias but high variance. One way out of this situation is to abandon the requirement of an unbiased estimator.

In the LS estimator, we have

$$\beta = (X^T X)^{-1} X^T Y$$

which could be improved by adding a small constant value λ to the diagonal entries of the matrix $X^T X$ before taking its inverse.

The result is the ridge regression estimator

$$\beta_{ridge} = (X^T X + \lambda I_p)^{-1} X^T Y$$

Ridge regression places a particular form of constraint on the parametersβ: β_{ridge} is chosen to minimize the penalized sum of squares:

$$\sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

which is equivalent to minimization of $\sum_{i=1}^{n} (y_i - x_i^T \beta)^2$ subject to $\lambda \sum_{j=1}^{p} \beta_j^2 \leq \gamma$.

Here we use $\lambda = 1$.

```
#ridge regression
ridge = linear_model.Ridge(alpha = 0.1)
ridge.fit (training[:,0:l], training[:,l])
res2 = ridge.predict(testing[:,0:l])
```

1.2.3. Tree

Decision Trees are a non-parametric supervised learning method, which can be used for regression. The goal is to create a model in a recursive binary splitting way that predicts the value of a target variable by learning simple decision rules inferred from the data features.

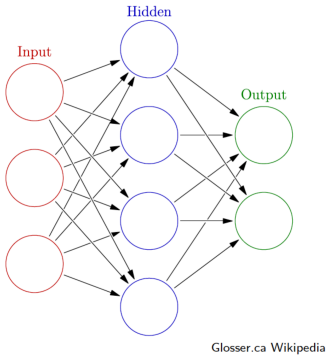
Recursive binary splitting is a Top-down and greedy approach. It begins at the top of the tree (at which point all observations belong to a single region) and then successively splits the predictor space. Each split is indicated via two new branches further down on the tree. At each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step [1].

```
#tree
from sklearn import tree
clf = tree.DecisionTreeRegressor()
clf = clf.fit(training[:,0:l], training[:,l])
res3 = clf.predict(testing[:,0:l])
```

1.2.4. Neural Networks (MLP)

A standard artificial neural network consists of many simple, connected "neurons", with each neuron has an activation function. Input neurons get activated through sensors perceiving the environment. Other neurons are activated through weighted connections from previously activated neurons.

Multilayer Perceptrons (MLPs) are multi-input and multi-output parametric functions de_ned by composing together many parametric functions. Each sub-function is known as a layer of the network, and each scalar layer output of one of these functions as a unit or feature. The number of units in each layer is called the width of a machine learning model, and the number of layers is called the depth [2].



II. Experiment Results

We test all these models on the four datasets and make prediction about total cost for each situation. The test error rate is shown in the following four tables and is obtained by the following equation:

$$\frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Knee	Ridge regression		Linear regression		Tree	
PCA for dimension reduction	Yes	No	Yes	No	Yes	No
Tree-based selection	0.266	0.271	0.266	0.271	0.239	0.256
Variance selection	0.263	0.267	0.263	0.267	0.287	0.274
No feature selection	0.329	0.264	0.329	0.264	0.343	0.298

Table 1. Prediction error rate for specialty hospital for knee surgery

Hip	Ridge regression		Linear regression		Tree	
PCA for dimension reduction	Yes	No	Yes	No	Yes	No
Tree-based selection	0.178	0.189	0.178	0.189	0.125	0.167
Variance selection	0.175	0.188	0.175	0.188	0.143	0.138
No feature selection	0.176	0.191	0.176	0.191	0.142	0.137

Table 2. Prediction error rate for specialty hospital for hip surgery

Knee	Ridge regression		Linear regression		Tree	
PCA for dimension reduction	Yes	No	Yes	No	Yes	No
Tree-based selection	0.224	0.209	0.224	0.209	0.239	0.206
Variance selection	0.221	0.207	0.221	0.207	0.265	0.239
No feature selection	0.221	0.207	0.221	0.207	0.269	0.238

Table 3. Prediction error rate for general hospital for knee surgery

Hip	Ridge regression		Linear regression		Tree	
PCA for dimension reduction	Yes	No	Yes	No	Yes	No
Tree-based selection	0.206	0.230	0.206	0.230	0.216	0.234
Variance selection	0.207	0.228	0.207	0.228	0.236	0.278
No feature selection	0.206	0.230	0.206	0.230	0.243	0.279

Table 4 Prediction error rate for general hospital for hip surgery

We also tested neural networks on Amazon AWS using theano, with learning rate = 0.01, batch size = 32, hidden layers = 2, and neurons in hidden layer = 1000, but the performance is very poor, with test error rate 81.25%.

```
[y1,y2] = test_mlp_parity(learning_rate=0.01, L1_reg=0.00, L2_reg=0.00, n_epochs=500,
             batch_size=32, n_hidden=1000, n_hiddenLayers=2,
             verbose=False)

... building the model
... training
Optimization complete.
Best validation score of 78.437500 % obtained at iteration 21, with test performance 81.250000 %

The training process for function test_mlp_parity ran for 0.33m
```

By looking at the test error rate of prediciton, we can see that feature selection can reduce the test error rate, especially that tree-based selection can significantly improve the performance of a tree. As for PCA, it can be used to reduce dimensions to avoid deviated data, but the size of training data will also be decreased thus the error rate may sometimes be incresed. Also, the differences between linear regression and ridge regression is very slight.

We choose tree-based feature selection and tree for regression to predict the total cost. With input data, we can recommend a type of hospital with the estimated cost for two types of hospitals.

```
INPUT DATA
Age:  50 to 69
Gender:  M
Race:  White
Length of Stay:  4
Type of Admission:  Elective
Patient Disposition:  Home w/ Home Health Services
CCS Diagnosis Code:  203
CCS Procedure Code:  153
APR Severity of Illness Code:  1
Predicted Charges for General Hospital:  [ 71541.]
Predicted Charges for Specialty Hospital:  [ 85928.5]
```

REFERENCES

[1] Giovanni Motta, “Data Mining” [Powerpoint slides], Nov 2015.

[2] Aurel A. Lazar. “Neural Networks and Deep Learning” [Powerpoint slides], Jan 2016.