ARTIFICIAL INTELLIGENCE CS561
ASSIGNMENT 1

Group 23

Manish Dash 140101087
Longkiri bey 140101035
M. Krishnananda Singh 174101005

Feature Selection:

All the features in a given dataset are not equally useful for the evaluation or performance of a ML model. Also a large number of features has a negative impact on the cost effectiveness of the model.

We want to select a “good” subset of features that produces near that of or even more than the results produced by including all the features.

Two simple choices for features that can be removed are :

- **Redundant:** These features are perfectly correlated with one or more features. So removing them does not reduce any performance.
- **Irrelevant**: Irrelevant features have no correlation and simply as noise. For example, serial numbers for the data elements. These can be removed as well.

Other than that we have to then carefully traverse the *state space* of the features to select the optimal subset of features. Consider **N** features in the data. If all possible subsets of the features are considered as nodes, we have $\mathbf{2^n - 1}$ nodes or *states*. We want to traverse this state space looking for the best subset of features.

We can evalute our subsets on the basis of evaluation scores. In our code we have used the Mean Squared Error of simple Linear Regression on publicly available datasets as the metric.

Feature selection can be done using two methods:

- Wrapper method
- Embedded method

We have implemented the wrapper method. As the name suggests, our feature selection algorithm acts as a wrapper around a ML model. At each iteration we select a subset of features and then evaluate that subset by training the model on it.

Datasets:

Our code can be run on any dataset presented in the csv format. More details are in the README.txt file.
We have tested on standard UCL machine learning datasets.

Technique 1: Sequential Forward Selection

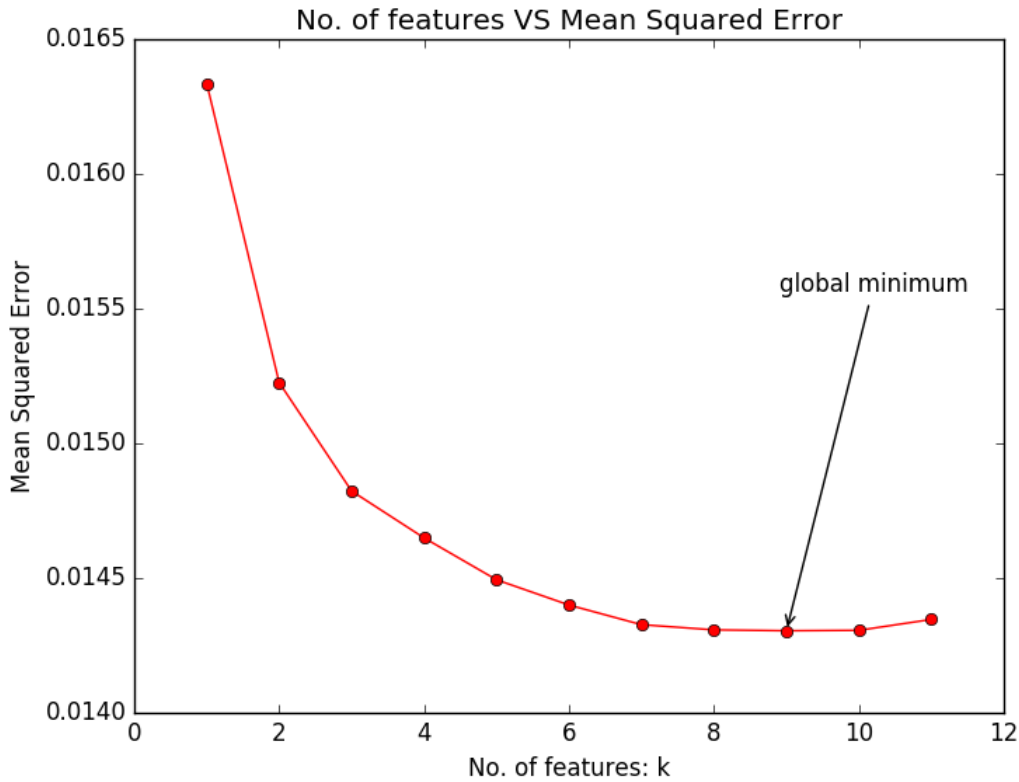
This algorithm greedily moves through the state-space looking for expanding the feature list only if an improvement is guaranteed.

1. We start with an empty list.
2. On each step, we want to extend the list from **n** features to **n+1**.
    - We check if there is any improvement by adding the non-included features into the current list one by one.
    - If there are one or more than one such features then the best feature (that improves the most) is included.
3. The algorithm goes on till we hit a maxima/minima (depending on criteria) or include all the features.

For visualizing, in the code we have run the process till we include all the features. This enables us to see the change in the MSE for each new feature included.

```
shadowfire ~/Documents/AI assignment 1 $ python part1.py
('iteration ', 1, ['"alcohol"'])
('iteration ', 2, ['"alcohol"', '"volatile acidity"'])
('iteration ', 3, ['"alcohol"', '"volatile acidity"', '"sulphates"'])
('iteration ', 4, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"'])
('iteration ', 5, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"'])
('iteration ', 6, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"'])
('iteration ', 7, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"'])
('iteration ', 8, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"', '"citri
c acid"'])
('iteration ', 9, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"', '"citri
c acid"', '"residual sugar"'])
('iteration ', 10, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"', '"citr
ic acid"', '"residual sugar"', '"density"'])
('iteration ', 11, ['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"', '"citr
ic acid"', '"residual sugar"', '"density"', '"fixed acidity"'])


The optimal subset of features is :
['"alcohol"', '"volatile acidity"', '"sulphates"', '"total sulfur dioxide"', '"chlorides"', '"pH"', '"free sulfur dioxide"']
```



Technique 2: Hill Climbing Technique

This algorithm is also greedy but it searches for the “local best” in both directions , i.e, both by adding a feature or removing a feature.

1. We start with a random subset of features as the current list.
2. At each iteration, we explore the neighbours in the state-space to look for the local best:
   - **i.** First we check the improvement (if any) by adding a new feature one by one. We note the *best entering feature* that improves the score(if any).
   - **ii.** We then check improvement by removing an existing feature one by one with replacement. We note the *best exiting feature* that improves the score (if any).
   - **iii.** We then compare the scores of **i.** and **ii.** Based on that we either extend the list or reduce it.
3. The algorithm ends when we hit a local maxima/minima. This happens when all the neighbours of the current state gives lesser scores. The algorithm stops at this point.

```
shadowfire ~/Documents/AI assignment 1 $ python part2.py
(6, 'Selected ', '"volatile acidity"', 0.014997841676127567)
(7, 'Selected ', '"alcohol"', 0.014534615674338076)
(8, 'Selected ', '"sulphates"', 0.014396612795895606)
(9, 'Selected ', '"pH"', 0.014325519175897944)
(10, 'Selected ', '"fixed acidity"', 0.014306057870095035)
(9, 'Removed ', '"citric acid"', 0.014304511294699255)
(10, 'Selected ', '"citric acid"', 0.014306057870095026)


The optimal subset of features is :
['"sulphates"', '"density"', '"free sulfur dioxide"', '"residual sugar"', '"fixed acidity"', '"alcohol"', '"pH"', '"total sulfur dioxide"', '"volatile
 acidity"']
```