Team Name: 1
Team members: Ziyang jiao, ukimalla, Huirou Zhou, FloraSun9101

**Application Project Final Report**

1.Introduction

In this project, our goal is to estimate forest cover type designation using basic descriptive information. In our dataset, there are twelve features, which includes Elevation in meters, Aspect in degrees azimuth, Slope in degrees ,Horz Dist to nearest surface water features, Vert Dist to nearest surface water features, Hillshade index at 9am, summer solstice, Hillshade index at noon, summer solstice, Hillshade index at 3pm, summer solstice, Soil_Type, Horz Dist to nearest wildfire ignition, Wilderness area designation. Most of our features are continuous and some of them are classification features. Thus, we implement some feature engineering techniques as following.

2.Data Engineering

2.1 One-Hot Encoding

For soil type feature, the first digit refers to the climatic zone; the second refers to the geologic. Because the last two digits have no special meaning, in this part, we extract climatic and geologic information, do the one hot encoding and add these two features to our original dataset and drop the soil type feature. For wilderness area feature, it has no unrelated information, so we do the dummy directly.

2.2 Feature importance analysis

We use the model.feature_importances_ to take a look at the importance of our different features, then we choose serval most important features (Elevation, Horizontal_Distance_To_Roadways, Horizontal_Distance_To_Fire_Points, horizontal distance to hydrology, etc) and then we make some combinations of them to get some new features. We integrate these new features with our original dataset to train our model. This part is implemented in feature_transform function. The following is the result of importance value based on this method.

```
Python Console
>>> runfile('/Users/ziyangjiao/Course/517/milestone2/milestone2.py',
Elevation 0.1982678902759039
Horizontal_Distance_To_Roadways 0.11207006044770279
Horizontal_Distance_To_Fire_Points 0.10548911028583266
Horizontal_Distance_To_Hydrology 0.06433172566028307
Vertical_Distance_To_Hydrology 0.06028788018591298
Aspect 0.054649425756805196
Hillshade_Noon 0.04964617886312048
Hillshade_9am 0.046120474359269276
Hillshade_3pm 0.045338798469144216
Slope 0.04193827919021227
climate_dummies_8 0.03657184068115587
area_dummies_Cache la Poudre 0.03588917643211216
climate_dummies_7 0.02752634611638876
climate_dummies_2 0.02689162208003467
climate_dummies_4 0.023806191341719483
geology_dummies_2 0.0209392493560747
area_dummies_Rawah 0.01516273547496725
area_dummies_Comanche Peak 0.013215236566847945
geology_dummies_7 0.012280285347149162
area_dummies_Neota 0.005194924932027069
```

Team Name: 1
Team members: Ziyang jiao, ukimalla, Huirou Zhou, FloraSun9101

2.3 Label Encode

We also encode our label to number value for our train, and we restore it after making prediction using label_encoder.inverse_transform() method.

3. Model Trained and Result

We tried several different models with different parameters and they are Neural network, Decision Trees and Linear Regression. The following table shows the result of accuracy for each model trained on our full dataset.

It is clearly that the decision trees model has the best training accuracy( ~90%), and Neural network model rank 2nd, the linear model has a poor performance.

| **Model Names** | **Neural Network with five layers** | **Neural Network with four layers** | **Linear Regression** | **Decision Tree** |
|---|---|---|---|---|
| Test 1 | 80% | 73% | 61% | 87% |
| Test 2 | 78% | 69% | 59% | 92% |
| Test 3 | 81% | 61% | 64% | 91% |
| Test 4 | 82% | 78% | 64% | 89% |
| Test 5 | 84% | 67% | 58% | 92% |
| Average | 81% | 69.6% | 61.2% | 90.2% |

Table 1: prediction accuracy of different models

Another table below is related to time consuming for our tested models. For this project, the most time-consuming model is the neural network(more than 7 mins with five layers) and when we try larger layers and nodes, it increases aggressively. The linear model and decision tree cost similar quantity of time but the former has a bad performance.

| **Model Names** | **Neural Network with five layers** | **Neural Network with four layers** | **Linear Regression** | **Decision Tree** |
|---|---|---|---|---|
| Test 1 | 513s | 348s | 87s | 93s |
| Test 2 | 492s | 413s | 64s | 102s |
| Test 3 | 468s | 322s | 72s | 79s |
| Test 4 | 537s | 303s | 95s | 88s |
| Test 5 | 488s | 358s | 101s | 97s |
| Average | 499.6s | 368.8s | 83.8s | 91.8s |

Table 2: time consumption of different models

Team Name: 1
Team members: Ziyang jiao, ukimalla, Huirou Zhou, FloraSun9101

## 4. Discussion

### 4.1 Model Flexibility

Model flexibility is evaluated on how models can fit various kinds of dataset. It is true that decision tree is not the best one because it is not good at regression dataset. The linear model has better flexibility on linear separated data but far less than Neural Network models, because it can fit almost any non-linear results.

### 4.2 Tradeoff between time and accuracy

For our project, the dataset is relatively small compared with reality, so we can focus on accuracy when test different models. The conclusion we get is limited to this project dataset. However, in real-word development, such as speech recognition or image processing, the training dataset becomes huge and sometimes we should compromise accuracy for better overall result.