DAT300 - Compulsory assignment 1

Group 9

Group name: Goofy

Members

- Joel Teklemariam
- Artush Mkrtchyan

Introduction

From what we have understood, the task is to make an ensemble model, a classification model and an ANN to predict forest types in different national parks. We are then going to compare the results and conclude which one was the best.

Our roles are exactly the same, because most of the time we have been sitting together to complete this assignment. That means both of us have coded and written text like this one.

Data pre-processing and visualisation

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.metrics import f1_score, confusion_matrix, roc_curve, auc
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from scipy import stats
from tensorflow.keras import models, layers
from tensorflow.keras.optimizers import Adam, Adamax, Adadelta
from tensorflow.keras.layers import LeakyReLU, PReLU, Dropout
from tensorflow.keras.callbacks import EarlyStopping

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
```

As seen above, we're importing what we think we may need for this project

In [ ]:

```
raw_data = pd.read_csv('/content/train.csv', index_col=0, delimiter=';') # Naming the tra
test_data = pd.read_csv('/content/test.csv', index_col=0, delimiter=';') # Naming the test
```

In [ ]:

```
print(raw_data.isnull().sum()) # Checking how many NaN there are
raw_data.head()
raw_data.info()
```

```
National Park                                                0
Elevation (meters)                                           0
Aspect (azimuth)                                             0
Slope (degrees)                                              0
Horizontal distance to water (meters)                        0
Vertical distance to water (meters)                          0
Horizontal distance to road (meters)                         0
Light at 9AM (hillshade)                                     0
Light at noon (hillshade)                                    0
Light at 3PM (hillshade)                                     0
Horizontal distance to fire ignition point (meters)          0
Soil 1                                                       0
Soil 2                                                       0
Soil 3                                                       0
Soil 4                                                       0
Soil 5                                                       0
Soil 6                                                       0
Soil 7                                                       0
Soil 8                                                       0
Soil 9                                                       0
Soil 10                                                      0
Soil 11                                                      0
Soil 12                                                      0
Soil 13                                                      0
Soil 14                                                      0
Soil 15                                                      0
Soil 16                                                      0
Soil 17                                                      0
Soil 18                                                      0
Soil 19                                                      0
Soil 20                                                      0
Soil 21                                                      0
Soil 22                                                      0
Soil 23                                                      0
Soil 24                                                      0
Soil 25                                                      0
Soil 26                                                      0
Soil 27                                                      0
Soil 28                                                      0
Soil 29                                                      0
Soil 30                                                      0
Soil 31                                                      0
Soil 32                                                      0
Soil 33                                                      0
Soil 34                                                      0
Soil 35                                                      0
Soil 36                                                      0
Soil 37                                                      0
Soil 38                                                      0
Soil 39                                                      0
Soil 40                                                      0
Forest type                                                  0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1398095 entries, 0 to 1398094
Data columns (total 52 columns):
 #   Column                                                Non-Null Count    Dtype
---  ------                                                --------------    -----
 0   National Park                                         1398095 non-null  object
 1   Elevation (meters)                                    1398095 non-null  int64
 2   Aspect (azimuth)                                      1398095 non-null  int64
 3   Slope (degrees)                                       1398095 non-null  int64
 4   Horizontal distance to water (meters)                 1398095 non-null  int64
 5   Vertical distance to water (meters)                   1398095 non-null  int64
 6   Horizontal distance to road (meters)                  1398095 non-null  int64
 7   Light at 9AM (hillshade)                              1398095 non-null  int64
```

```
 8   Light at noon (hillshade)                                   1398095 non-null  int64
 9   Light at 3PM (hillshade)                                    1398095 non-null  int64
 10  Horizontal distance to fire ignition point (meters)        1398095 non-null  int64
 11  Soil 1                                                      1398095 non-null  int64
 12  Soil 2                                                      1398095 non-null  int64
 13  Soil 3                                                      1398095 non-null  int64
 14  Soil 4                                                      1398095 non-null  int64
 15  Soil 5                                                      1398095 non-null  int64
 16  Soil 6                                                      1398095 non-null  int64
 17  Soil 7                                                      1398095 non-null  int64
 18  Soil 8                                                      1398095 non-null  int64
 19  Soil 9                                                      1398095 non-null  int64
 20  Soil 10                                                     1398095 non-null  int64
 21  Soil 11                                                     1398095 non-null  int64
 22  Soil 12                                                     1398095 non-null  int64
 23  Soil 13                                                     1398095 non-null  int64
 24  Soil 14                                                     1398095 non-null  int64
 25  Soil 15                                                     1398095 non-null  int64
 26  Soil 16                                                     1398095 non-null  int64
 27  Soil 17                                                     1398095 non-null  int64
 28  Soil 18                                                     1398095 non-null  int64
 29  Soil 19                                                     1398095 non-null  int64
 30  Soil 20                                                     1398095 non-null  int64
 31  Soil 21                                                     1398095 non-null  int64
 32  Soil 22                                                     1398095 non-null  int64
 33  Soil 23                                                     1398095 non-null  int64
 34  Soil 24                                                     1398095 non-null  int64
 35  Soil 25                                                     1398095 non-null  int64
 36  Soil 26                                                     1398095 non-null  int64
 37  Soil 27                                                     1398095 non-null  int64
 38  Soil 28                                                     1398095 non-null  int64
 39  Soil 29                                                     1398095 non-null  int64
 40  Soil 30                                                     1398095 non-null  int64
 41  Soil 31                                                     1398095 non-null  int64
 42  Soil 32                                                     1398095 non-null  int64
 43  Soil 33                                                     1398095 non-null  int64
 44  Soil 34                                                     1398095 non-null  int64
 45  Soil 35                                                     1398095 non-null  int64
 46  Soil 36                                                     1398095 non-null  int64
 47  Soil 37                                                     1398095 non-null  int64
 48  Soil 38                                                     1398095 non-null  int64
 49  Soil 39                                                     1398095 non-null  int64
 50  Soil 40                                                     1398095 non-null  int64
 51  Forest type                                                 1398095 non-null  object
dtypes: int64(50), object(2)
memory usage: 565.3+ MB
```

In [ ]:

```
print(test_data.isnull().sum()) # Checking how many NaN there are
test_data.head()
test_data.info()
```

```
National Park                                          0
Elevation (meters)                                     0
Aspect (azimuth)                                       0
Slope (degrees)                                        0
Horizontal distance to water (meters)                  0
Vertical distance to water (meters)                    0
Horizontal distance to road (meters)                   0
Light at 9AM (hillshade)                               0
Light at noon (hillshade)                              0
Light at 3PM (hillshade)                               0
Horizontal distance to fire ignition point (meters)    0
Soil 1                                                 0
Soil 2                                                 0
Soil 3                                                 0
```

```
Soil 4                                                     0
Soil 5                                                     0
Soil 6                                                     0
Soil 7                                                     0
Soil 8                                                     0
Soil 9                                                     0
Soil 10                                                    0
Soil 11                                                    0
Soil 12                                                    0
Soil 13                                                    0
Soil 14                                                    0
Soil 15                                                    0
Soil 16                                                    0
Soil 17                                                    0
Soil 18                                                    0
Soil 19                                                    0
Soil 20                                                    0
Soil 21                                                    0
Soil 22                                                    0
Soil 23                                                    0
Soil 24                                                    0
Soil 25                                                    0
Soil 26                                                    0
Soil 27                                                    0
Soil 28                                                    0
Soil 29                                                    0
Soil 30                                                    0
Soil 31                                                    0
Soil 32                                                    0
Soil 33                                                    0
Soil 34                                                    0
Soil 35                                                    0
Soil 36                                                    0
Soil 37                                                    0
Soil 38                                                    0
Soil 39                                                    0
Soil 40                                                    0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 599184 entries, 0 to 599183
Data columns (total 51 columns):
 #   Column                                                Non-Null Count   Dtype
---  ------                                                --------------   -----
 0   National Park                                         599184 non-null  object
 1   Elevation (meters)                                    599184 non-null  int64
 2   Aspect (azimuth)                                      599184 non-null  int64
 3   Slope (degrees)                                       599184 non-null  int64
 4   Horizontal distance to water (meters)                 599184 non-null  int64
 5   Vertical distance to water (meters)                   599184 non-null  int64
 6   Horizontal distance to road (meters)                  599184 non-null  int64
 7   Light at 9AM (hillshade)                              599184 non-null  int64
 8   Light at noon (hillshade)                             599184 non-null  int64
 9   Light at 3PM (hillshade)                              599184 non-null  int64
 10  Horizontal distance to fire ignition point (meters)  599184 non-null  int64
 11  Soil 1                                                599184 non-null  int64
 12  Soil 2                                                599184 non-null  int64
 13  Soil 3                                                599184 non-null  int64
 14  Soil 4                                                599184 non-null  int64
 15  Soil 5                                                599184 non-null  int64
 16  Soil 6                                                599184 non-null  int64
 17  Soil 7                                                599184 non-null  int64
 18  Soil 8                                                599184 non-null  int64
 19  Soil 9                                                599184 non-null  int64
 20  Soil 10                                               599184 non-null  int64
 21  Soil 11                                               599184 non-null  int64
 22  Soil 12                                               599184 non-null  int64
```

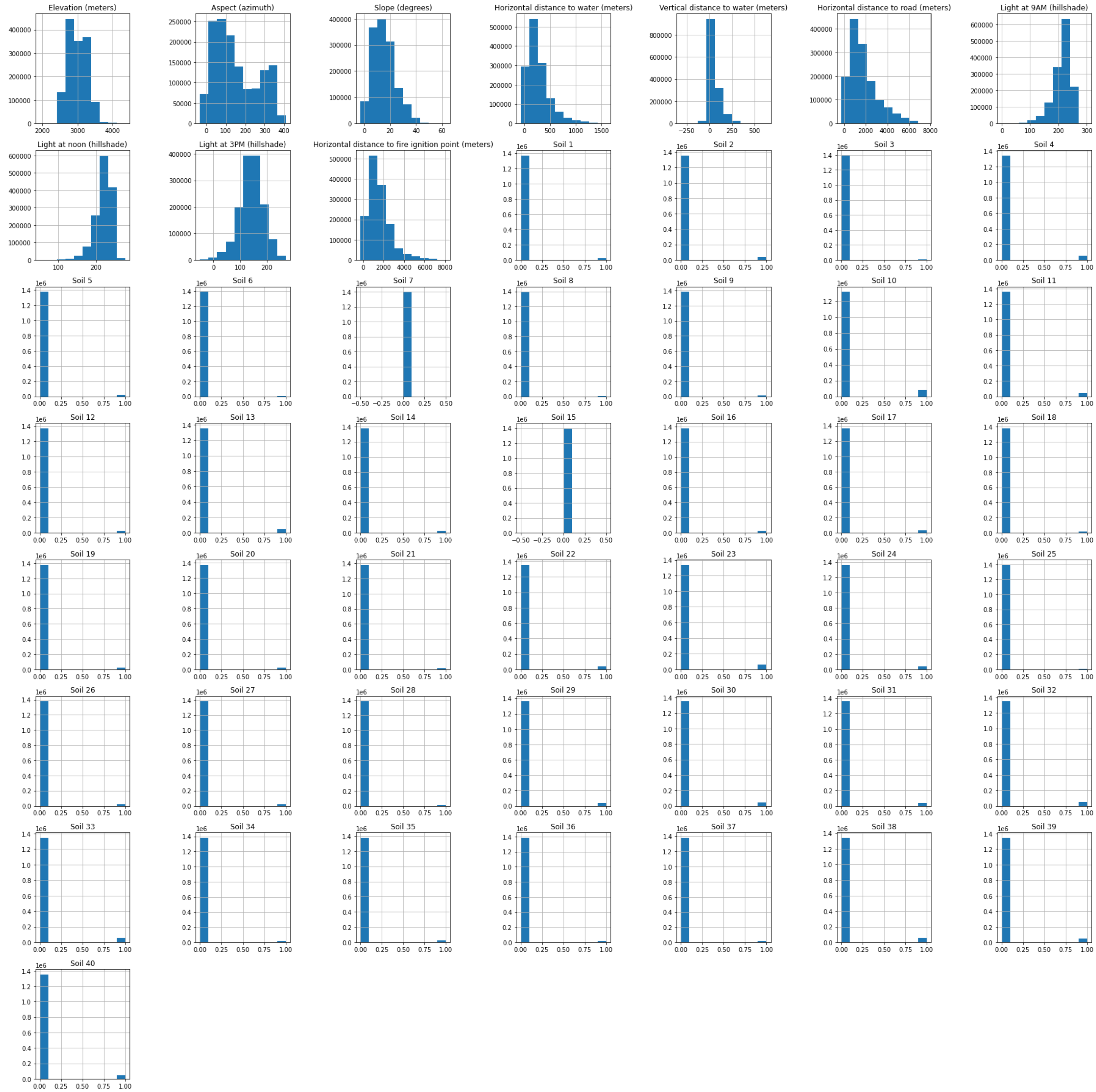
```
 23   Soil 13                                                  599184 non-null  int64
 24   Soil 14                                                  599184 non-null  int64
 25   Soil 15                                                  599184 non-null  int64
 26   Soil 16                                                  599184 non-null  int64
 27   Soil 17                                                  599184 non-null  int64
 28   Soil 18                                                  599184 non-null  int64
 29   Soil 19                                                  599184 non-null  int64
 30   Soil 20                                                  599184 non-null  int64
 31   Soil 21                                                  599184 non-null  int64
 32   Soil 22                                                  599184 non-null  int64
 33   Soil 23                                                  599184 non-null  int64
 34   Soil 24                                                  599184 non-null  int64
 35   Soil 25                                                  599184 non-null  int64
 36   Soil 26                                                  599184 non-null  int64
 37   Soil 27                                                  599184 non-null  int64
 38   Soil 28                                                  599184 non-null  int64
 39   Soil 29                                                  599184 non-null  int64
 40   Soil 30                                                  599184 non-null  int64
 41   Soil 31                                                  599184 non-null  int64
 42   Soil 32                                                  599184 non-null  int64
 43   Soil 33                                                  599184 non-null  int64
 44   Soil 34                                                  599184 non-null  int64
 45   Soil 35                                                  599184 non-null  int64
 46   Soil 36                                                  599184 non-null  int64
 47   Soil 37                                                  599184 non-null  int64
 48   Soil 38                                                  599184 non-null  int64
 49   Soil 39                                                  599184 non-null  int64
 50   Soil 40                                                  599184 non-null  int64
dtypes: int64(50), object(1)
memory usage: 237.7+ MB
```

What we found out is that this dataset does not contain any NaN values

In [ ]:

```
# Histograms below

raw_data.hist(figsize=(25, 25))
plt.tight_layout()
plt.show()
```



We can see that there are no instances of soil type 7 and 15. Furthermore, one can see that most of the data is skewed, which can affect the results later.

In [ ]:

```
# Removing the columns soil 7 and soil 15

raw_data.drop(columns=["Soil 7", "Soil 15"], inplace=True)
test_data.drop(columns=["Soil 7", "Soil 15"], inplace=True)
```

In [ ]:

```
raw_data.iloc[:,0:11].head()
le = LabelEncoder()

dummies = pd.get_dummies(raw_data["National Park"], drop_first=True)
raw_data["Forest type"] = le.fit_transform(raw_data["Forest type"])

enc_data = pd.concat([raw_data, dummies], axis=1)
enc_data = enc_data._get_numeric_data()

enc_data.head()
```

Out[ ]:

	Elevation (meters)	Aspect (azimuth)	Slope (degrees)	Horizontal distance to water (meters)	Vertical distance to water (meters)	Horizontal distance to road (meters)	Light at 9AM (hillshade)	Light at noon (hillshade)	Light at 3PM (hillshade)	Horizontal distance to fire ignition point (meters)
0	3204	65	22	55	83	566	195	214	128	1814
1	2688	309	34	278	61	224	250	223	44	128
2	3137	267	13	149	37	3146	190	256	165	1075
3	3286	300	7	275	56	147	241	167	155	-134
4	2864	82	3	221	136	2184	202	250	185	1287

5 rows × 52 columns

In [ ]:

```
test_data.iloc[:,0:11].head()
le = LabelEncoder()

dummies_test = pd.get_dummies(test_data["National Park"], drop_first=True)
enc_test = pd.concat([test_data, dummies_test], axis=1) # Merging / concatenating two data
enc_test = enc_test._get_numeric_data()
test_data.head()
```

Out[ ]:

	National Park	Elevation (meters)	Aspect (azimuth)	Slope (degrees)	Horizontal distance to water (meters)	Vertical distance to water (meters)	Horizontal distance to road (meters)	Light at 9AM (hillshade)	Light at noon (hillshade)	Light at 3PM (hillshade)	..
0	Mount Rainer	2590	128	32	32	7	507	219	227	182	..
1	Mount Rainer	3107	112	18	961	2	1168	179	215	101	..
2	Mount Rainer	3018	280	10	394	16	1922	233	235	149	..
3	Mount Rainer	3268	73	7	485	81	3463	238	211	135	..
4	Mount Rainer	3474	340	13	815	47	1848	241	193	148	..

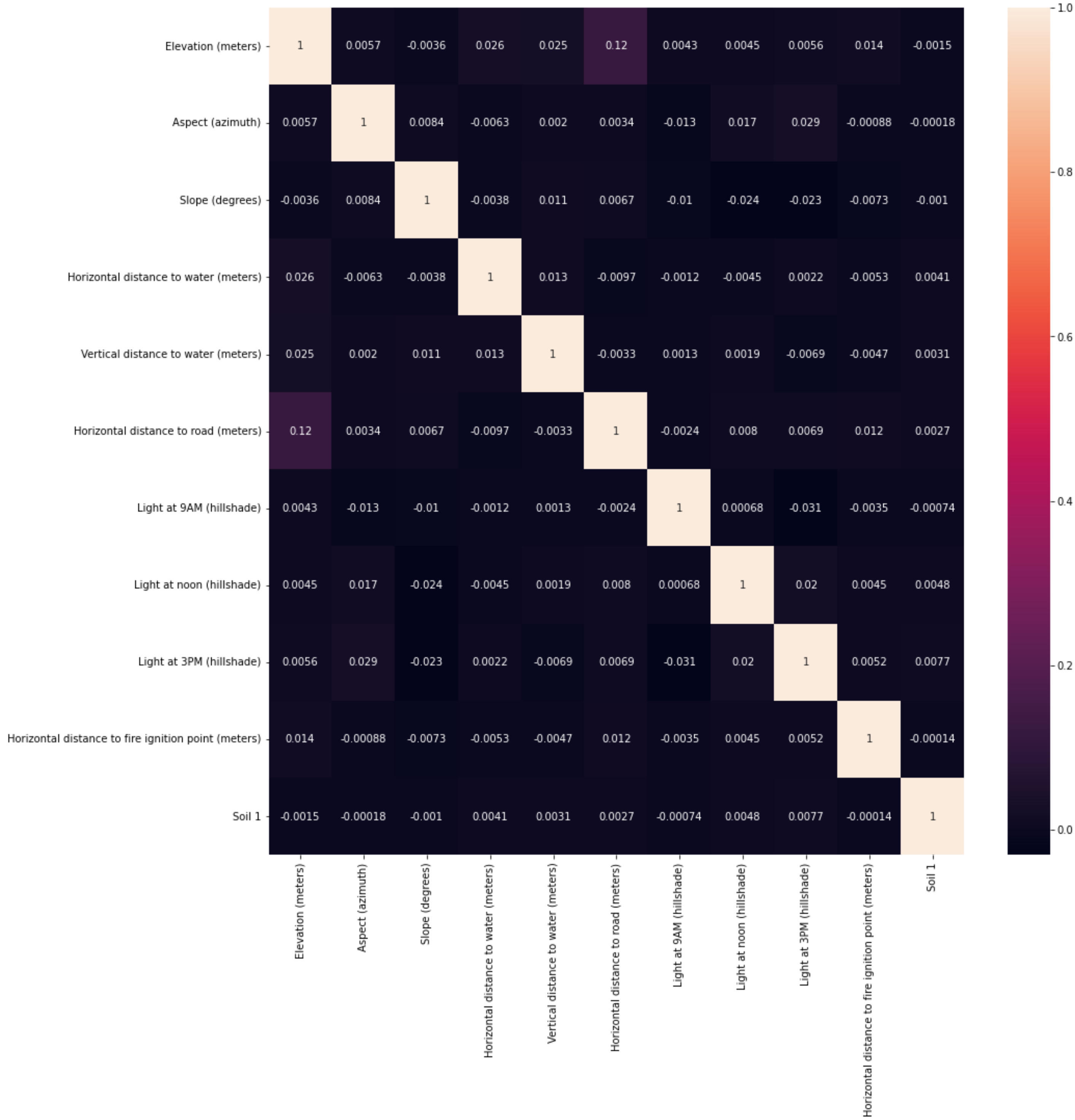
5 rows × 49 columns

We are making dummies out of the categorical column "National Park" and later concatenating the dummies with the original dataframe.

In [ ]:

```
plt.figure(figsize=(15, 15)) # Making it big for easier inspection
sliced_data = enc_data.iloc[:,0:11]
corr_matrix1 = sliced_data.corr()

sns.heatmap(corr_matrix1, annot=True)
plt.show()
```

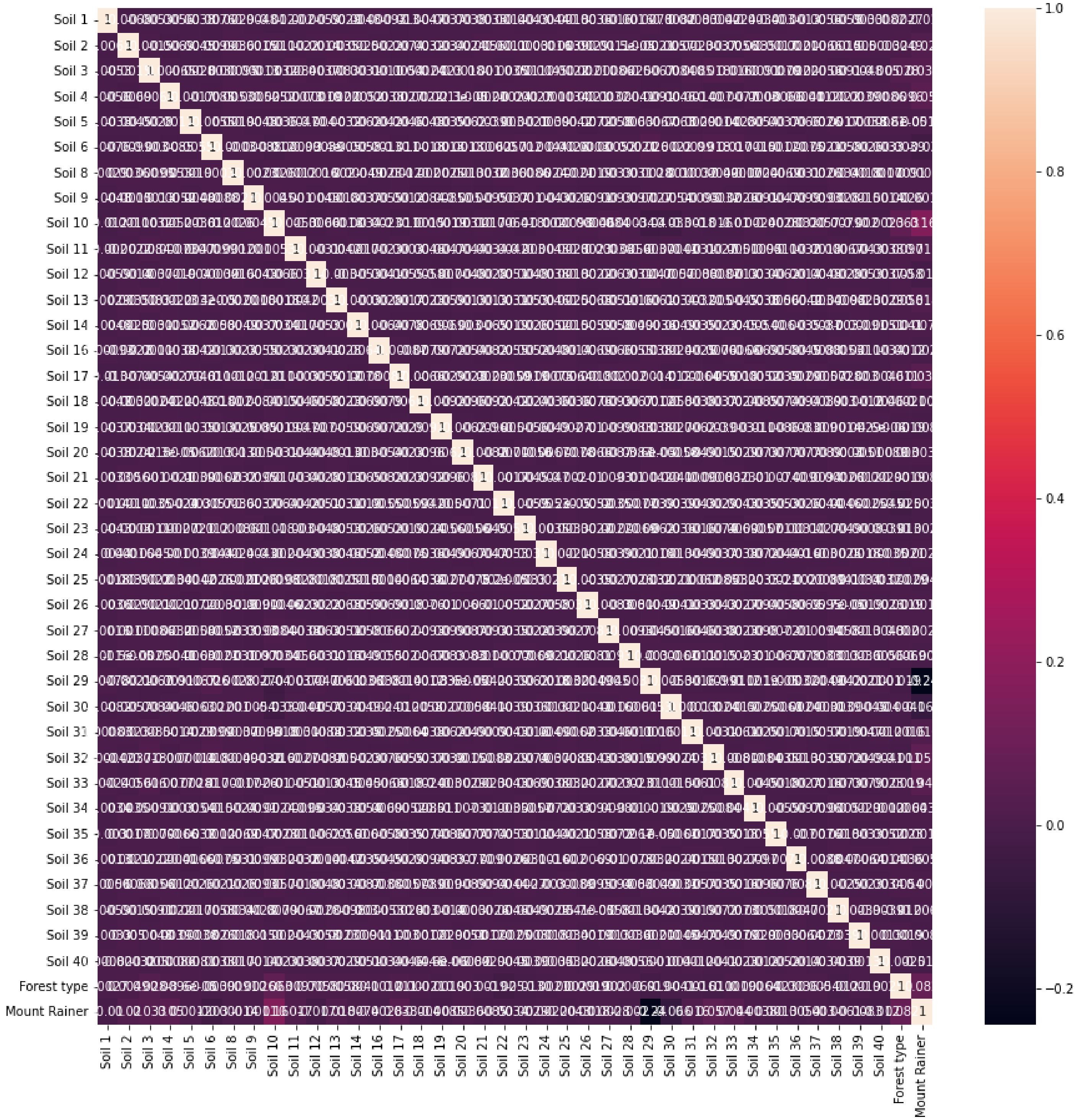


In [ ]:

```
plt.figure(figsize=(15, 15)) # Making it big for easier inspection
sliced_data = enc_data.iloc[:,10:50]
corr_matrix1 = sliced_data.corr()

sns.heatmap(corr_matrix1, annot=True)
plt.show()


# there is almost no correlation between the soils.
```



As mentioned under the histograms, there are no instances of soil 7 and soil 15. This is further visualised in the heatmap shown above. Furthermore, one can see from the heatmap that the soils have little to no correleation between them.

Modelling

1. Scikit-learn

To be honest, we had more problems with our SCI-kit models than with our ANN model. That is because we tried to do a grid search on Gradient Boosting Classifier and SVC. Since the dataset is enormous, the process of training the models took several hours. That is why we removed grid search altogether and even changed which models we use. Therefore, we use Logistic Regression instead of SVC and Random Forest instead of Gradient Boosting Classifier.

Furthermore, we did not test the Random Forest on all the data as a whole, because the model did not stop running even after three hours. Additionally, we just want to mention that all the models did relatively well on the same problem in terms of accuracy. Only difference is that our ANN model was on average faster than our shallow learning models when we tried to tune them. Without tuning, our shallow learning models were faster than our ANN model.

Classification model based on algorithms from scikit-learn (e.g. logistic regression, possibly regularized, Support Vector Classifier, etc.).

In [ ]:

```
# Code for scikit-learn based model

X = enc_data.drop(["Forest type"], axis=1)
y = enc_data["Forest type"]
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X,y, test_size=0.3, random_state=21)

print("X_train: ",X_train.shape)
print("X_test: ",X_test.shape)
print("y_train: ",y_train.shape)
print("y_test: ",y_test.shape)
```

```
X_train:  (978666, 51)
X_test:  (419429, 51)
y_train:  (978666,)
y_test:  (419429,)
```

Scaling the data below for use in logistic regression

In [ ]:

```
# Scaling the data with StandardScaler()

sc = StandardScaler()
sc.fit(X_train)


# Transform (standardise) both X_train and X_test with mean and STD from
# training data

X_train_sc = sc.transform(X_train)
X_test_sc = sc.transform(X_test)
```

In [ ]:

```
lr = make_pipeline(
    StandardScaler(),
    LogisticRegression(random_state=21,
                       n_jobs=-1),).fit(X_train_sc, y_train)

lr.predict(X_test_sc)



print('Logistic Regression training data accuracy: {0:.4f}'.format(
    lr.score(X_train_sc, y_train)))

print('Logistic Regression test data accuracy: {0:.4f}'.format(
    lr.score(X_test_sc, y_test)))
```

```
Logistic Regression training data accuracy: 0.9455
Logistic Regression test data accuracy: 0.9452
```

Ensemble model (e.g. random forest or similar)

Ensemble model of Random Forest Classifier

In [ ]:
```
rand = RandomForestClassifier(random_state=21,
                                n_estimators= 100,
                                max_features= "auto"
                                )
```

In [ ]:
```
rand.fit(X_train, y_train) # training the random forest
# Should have trained with the whole data X and y but it took too much time
```

Out[ ]:
```
RandomForestClassifier(random_state=21)
```

In [ ]:
```
print('Random forest training data accuracy: {0:.4f}'.format(
    rand.score(X_train, y_train)))

print('Random forest test data accuracy: {0:.4f}'.format(
    rand.score(X_test, y_test)))
```

```
Random forest training data accuracy: 1.0000
Random forest test data accuracy: 0.9740
```

2. Neural Network with Keras

When we started we made the mistake of not using the sigmoid activation function nor the binary crossentropy loss function. Because of this, our results started out quite bad, but after realizing this we quickly fixed it. We have also tried several different activation functions like PreLU and Swish, but quickly settled with ELU. In terms of optimizers, we found out that the standard Adam works best. Much better than Adagrad and Adadelta, but only slightly better than rmsprop. We also started with few neurons, but quickly found out that using big handfuls of neurons was the best option.

We used many attempts to get above the beat me since we could not get past 96% for the first few days. After a while we managed that by using ELU starting with 1024 neurons and halving the number for each layer. All in all, this task required a lot of trial and error in terms of tuning the ANN model.

In [66]:
```
# Code for creating and training a ANN with Keras

model = models.Sequential([
    layers.Dense(1024, activation ="ELU"),
    layers.Dense(512, activation = "ELU"),
    layers.Dense(256, activation = "ELU"),
    layers.Dense(128, activation = "ELU"),
    layers.Dense(64, activation = "ELU"),
    layers.Dense(1, activation = "sigmoid")])

# Compile model
model.compile(optimizer=Adam(learning_rate=0.0005),
              loss='binary_crossentropy',
              metrics=['accuracy'])

callback = EarlyStopping(monitor='loss', patience=5)

# Fit model (in the same manner as you would with scikit-learn)
model.fit(X_train,
          y_train,
          epochs=100,
```

```
            callbacks= callback,
            batch_size=160,
            validation_split=0.4)
```

```
Epoch 1/100
3670/3670 [==============================] - 19s 5ms/step - loss: 0.3854 - accuracy: 0.883
9 - val_loss: 0.1831 - val_accuracy: 0.9215
Epoch 2/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.1532 - accuracy: 0.937
5 - val_loss: 0.1219 - val_accuracy: 0.9499
Epoch 3/100
3670/3670 [==============================] - 21s 6ms/step - loss: 0.1220 - accuracy: 0.948
3 - val_loss: 0.1063 - val_accuracy: 0.9546
Epoch 4/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.1072 - accuracy: 0.954
4 - val_loss: 0.0863 - val_accuracy: 0.9648
Epoch 5/100
3670/3670 [==============================] - 20s 5ms/step - loss: 0.1016 - accuracy: 0.956
5 - val_loss: 0.1249 - val_accuracy: 0.9438
Epoch 6/100
3670/3670 [==============================] - 20s 6ms/step - loss: 0.0954 - accuracy: 0.959
6 - val_loss: 0.0809 - val_accuracy: 0.9661
Epoch 7/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0911 - accuracy: 0.961
1 - val_loss: 0.0782 - val_accuracy: 0.9668
Epoch 8/100
3670/3670 [==============================] - 20s 5ms/step - loss: 0.0910 - accuracy: 0.961
1 - val_loss: 0.0916 - val_accuracy: 0.9626
Epoch 9/100
3670/3670 [==============================] - 19s 5ms/step - loss: 0.0878 - accuracy: 0.962
8 - val_loss: 0.0881 - val_accuracy: 0.9621
Epoch 10/100
3670/3670 [==============================] - 19s 5ms/step - loss: 0.0867 - accuracy: 0.963
1 - val_loss: 0.1062 - val_accuracy: 0.9534
Epoch 11/100
3670/3670 [==============================] - 24s 7ms/step - loss: 0.0857 - accuracy: 0.963
4 - val_loss: 0.0883 - val_accuracy: 0.9609
Epoch 12/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0841 - accuracy: 0.964
1 - val_loss: 0.0754 - val_accuracy: 0.9683
Epoch 13/100
3670/3670 [==============================] - 21s 6ms/step - loss: 0.0831 - accuracy: 0.964
6 - val_loss: 0.0810 - val_accuracy: 0.9648
Epoch 14/100
3670/3670 [==============================] - 24s 7ms/step - loss: 0.0816 - accuracy: 0.965
3 - val_loss: 0.0701 - val_accuracy: 0.9711
Epoch 15/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0820 - accuracy: 0.964
9 - val_loss: 0.0836 - val_accuracy: 0.9638
Epoch 16/100
3670/3670 [==============================] - 20s 6ms/step - loss: 0.0803 - accuracy: 0.965
8 - val_loss: 0.0727 - val_accuracy: 0.9687
Epoch 17/100
3670/3670 [==============================] - 21s 6ms/step - loss: 0.0801 - accuracy: 0.965
8 - val_loss: 0.0723 - val_accuracy: 0.9707
Epoch 18/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0802 - accuracy: 0.965
9 - val_loss: 0.0815 - val_accuracy: 0.9649
Epoch 19/100
3670/3670 [==============================] - 21s 6ms/step - loss: 0.0792 - accuracy: 0.966
5 - val_loss: 0.0724 - val_accuracy: 0.9689
Epoch 20/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0786 - accuracy: 0.966
5 - val_loss: 0.0803 - val_accuracy: 0.9659
Epoch 21/100
```

```
3670/3670 [==============================] - 21s 6ms/step - loss: 0.0780 - accuracy: 0.966
7 - val_loss: 0.0698 - val_accuracy: 0.9707
Epoch 22/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0780 - accuracy: 0.966
8 - val_loss: 0.0846 - val_accuracy: 0.9639
Epoch 23/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0780 - accuracy: 0.966
9 - val_loss: 0.0788 - val_accuracy: 0.9657
Epoch 24/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0773 - accuracy: 0.967
1 - val_loss: 0.0693 - val_accuracy: 0.9716
Epoch 25/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0757 - accuracy: 0.967
9 - val_loss: 0.0761 - val_accuracy: 0.9684
Epoch 26/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0769 - accuracy: 0.967
2 - val_loss: 0.0676 - val_accuracy: 0.9719
Epoch 27/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0768 - accuracy: 0.967
2 - val_loss: 0.0681 - val_accuracy: 0.9717
Epoch 28/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0758 - accuracy: 0.967
6 - val_loss: 0.0749 - val_accuracy: 0.9677
Epoch 29/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0753 - accuracy: 0.968
0 - val_loss: 0.0724 - val_accuracy: 0.9699
Epoch 30/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0756 - accuracy: 0.968
0 - val_loss: 0.0718 - val_accuracy: 0.9699
Epoch 31/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0752 - accuracy: 0.967
8 - val_loss: 0.0689 - val_accuracy: 0.9704
Epoch 32/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0747 - accuracy: 0.968
4 - val_loss: 0.0759 - val_accuracy: 0.9687
Epoch 33/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0744 - accuracy: 0.968
2 - val_loss: 0.0871 - val_accuracy: 0.9619
Epoch 34/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0743 - accuracy: 0.968
3 - val_loss: 0.0679 - val_accuracy: 0.9714
Epoch 35/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0738 - accuracy: 0.968
4 - val_loss: 0.0695 - val_accuracy: 0.9700
Epoch 36/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0737 - accuracy: 0.968
6 - val_loss: 0.0863 - val_accuracy: 0.9622
Epoch 37/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0734 - accuracy: 0.968
7 - val_loss: 0.0693 - val_accuracy: 0.9707
Epoch 38/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0736 - accuracy: 0.968
6 - val_loss: 0.0758 - val_accuracy: 0.9673
Epoch 39/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0727 - accuracy: 0.968
8 - val_loss: 0.0692 - val_accuracy: 0.9708
Epoch 40/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0735 - accuracy: 0.968
6 - val_loss: 0.0635 - val_accuracy: 0.9732
Epoch 41/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0721 - accuracy: 0.969
2 - val_loss: 0.0768 - val_accuracy: 0.9670
Epoch 42/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0731 - accuracy: 0.968
7 - val_loss: 0.0708 - val_accuracy: 0.9703
Epoch 43/100
```

```
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0722 - accuracy: 0.969
4 - val_loss: 0.0642 - val_accuracy: 0.9726
Epoch 44/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0726 - accuracy: 0.969
2 - val_loss: 0.0730 - val_accuracy: 0.9685
Epoch 45/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0725 - accuracy: 0.969
1 - val_loss: 0.0803 - val_accuracy: 0.9648
Epoch 46/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0718 - accuracy: 0.969
3 - val_loss: 0.0666 - val_accuracy: 0.9726
Epoch 47/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0724 - accuracy: 0.969
1 - val_loss: 0.0824 - val_accuracy: 0.9645
Epoch 48/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0720 - accuracy: 0.969
3 - val_loss: 0.0668 - val_accuracy: 0.9726
Epoch 49/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0711 - accuracy: 0.969
5 - val_loss: 0.0657 - val_accuracy: 0.9724
Epoch 50/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0710 - accuracy: 0.969
5 - val_loss: 0.0630 - val_accuracy: 0.9738
Epoch 51/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0710 - accuracy: 0.969
8 - val_loss: 0.0634 - val_accuracy: 0.9731
Epoch 52/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0713 - accuracy: 0.969
5 - val_loss: 0.1030 - val_accuracy: 0.9550
Epoch 53/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0711 - accuracy: 0.969
5 - val_loss: 0.0682 - val_accuracy: 0.9710
Epoch 54/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0703 - accuracy: 0.969
9 - val_loss: 0.0741 - val_accuracy: 0.9685
Epoch 55/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0714 - accuracy: 0.969
3 - val_loss: 0.0714 - val_accuracy: 0.9699
Epoch 56/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0704 - accuracy: 0.970
0 - val_loss: 0.0722 - val_accuracy: 0.9687
Epoch 57/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0704 - accuracy: 0.970
0 - val_loss: 0.0807 - val_accuracy: 0.9653
Epoch 58/100
3670/3670 [==============================] - 17s 5ms/step - loss: 0.0708 - accuracy: 0.969
6 - val_loss: 0.0764 - val_accuracy: 0.9670
Epoch 59/100
3670/3670 [==============================] - 18s 5ms/step - loss: 0.0704 - accuracy: 0.969
9 - val_loss: 0.0730 - val_accuracy: 0.9684
```

Out[66]:

```
<keras.callbacks.History at 0x7f7b01d72a50>
```

In [67]:

```python
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 1024)              53248

 dense_13 (Dense)            (None, 512)               524800

 dense_14 (Dense)            (None, 256)               131328
```

```
 dense_15 (Dense)            (None, 128)               32896

 dense_16 (Dense)            (None, 64)                8256

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 750,593
Trainable params: 750,593
Non-trainable params: 0
_________________________________________________________________
```

In [68]:

```
predictions = model.predict(enc_test)
```

In [69]:

```
model.evaluate(X_test, y_test)
```

```
13108/13108 [==============================] - 29s 2ms/step - loss: 0.0730 - accuracy: 0.9
681
```

Out[69]:

```
[0.07301066070795059, 0.9680947065353394]
```

ANN Results

Report on your best ANN found and print out relevant metrics

In [70]:

```
submission_df = pd.DataFrame(data=list(range(len(predictions))),
                             columns=["Index"])

submission_df["Predicted"] = predictions
submission_df = submission_df.round(0)

submission_df['Predicted'] = np.where(submission_df['Predicted'] == 1,
                                      "Lodgepole", "Cottonwood")

submission_df.to_csv("CA1_goofy_submission4.csv", index=False)
submission_df
```

Out[70]:

	Index	Predicted
0	0	Lodgepole
1	1	Lodgepole
2	2	Lodgepole
3	3	Cottonwood
4	4	Cottonwood
...	...	...
599179	599179	Lodgepole
599180	599180	Lodgepole
599181	599181	Lodgepole
599182	599182	Cottonwood
599183	599183	Lodgepole

599184 rows × 2 columns

Discussion / conclusion

Provide a summary of the assignment: (you are required to address **the first three** points of the list below)

- obstacles / problems you have met regarding the modelling proces
- degree of success of the three models
- given more time, what would be done differently
- further comments (if any)

We had problems running the ensemble classifier, especially when using the support vector classifier. The main problem was the amount of time it took to run the code for the whole training data. Later we decided to use a logistic regressor to classify. We also had to preprocess the data differently than we initially had planned. We first used the LabelEncoder for categorical data in column National Park, but then changed it to dummies. Given more time, we would experiment with other activation and loss functions. We had some ideas to make the model more complex without overfitting it, as in adding more layers and implement more neurons to the layers.

To conclude, our models were all quite accurate on predicting with an average of about 97%. The only difference were how fast the models were at training, where the shallow learning models did not hold well against our ANN model when doing grid searches.