Advanced Data Science Capstone

By Theo Thomas

Use Case

- I decided to use the Forest Cover Type prediction as I used in Kaggle training.

  In the training I used RandomForest Classifier.
  I now want to see if I can improve results using DeepLearning/Keras.
- See also
  - https://www.kaggle.com/c/learn-together/data
  - https://www.kaggle.com/uciml/forest-cover-type-dataset

Dataset

- The dataset is well described at the source
- According to https://archive.ics.uci.edu/ml/datasets/covertype which is the origin of the data the features are defined as follows
- Name / Data Type / Measurement / Description
- Elevation / quantitative /meters / Elevation in meters
  Aspect / quantitative / azimuth / Aspect in degrees azimuth
  Slope / quantitative / degrees / Slope in degrees
  Horizontal_Distance_To_Hydrology / quantitative / meters / Horz Dist to nearest surface water features
  Vertical_Distance_To_Hydrology / quantitative / meters / Vert Dist to nearest surface water features
  Horizontal_Distance_To_Roadways / quantitative / meters / Horz Dist to nearest roadway
  Hillshade_9am / quantitative / 0 to 255 index / Hillshade index at 9am, summer solstice
  Hillshade_Noon / quantitative / 0 to 255 index / Hillshade index at noon, summer soltice
  Hillshade_3pm / quantitative / 0 to 255 index / Hillshade index at 3pm, summer solstice
  Horizontal_Distance_To_Fire_Points / quantitative / meters / Horz Dist to nearest wildfire ignition points
  Wilderness_Area (4 binary columns) / qualitative / 0 (absence) or 1 (presence) / Wilderness area designation
  Soil_Type (40 binary columns) / qualitative / 0 (absence) or 1 (presence) / Soil Type designation
  Cover_Type (7 types) / integer / 1 to 7 / Forest Cover Type designation
- All features are numeric. Wilderness_AreaX and Soil_TypeX are binary features of categorical variables.

Data Quality (1)

- Since the dataset is used for many trainings and is meant to be used for machine learning training without having to do much cleansing the data quality is high.
- There are for example no NAN’s in the dataset.

Data Quality (2)

- The binary variables are in a One hot encoded format

Check Soil OHE

In [152]:
```
soil_columns = X.loc[:, 'Soil_Type1':'Soil_Type40']
soil_columns['check'] = soil_columns.sum(axis=1) # let's see if we have no missing values or duplicates
soil_columns
```

Out[152]:

Id	Soil_Type1	Soil_Type2	Soil_Type3	Soil_Type4	Soil_Type5	Soil_Type6	Soil_Type7	Soil_Type8	Soil_Type9	Soil_Type10	...	Soil_Type32	Soil_Type33	Soi
1	0	0	0	0	0	0	0	0	0	0	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	
5	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
15116	0	0	0	1	0	0	0	0	0	0	...	0	0	
15117	0	0	0	1	0	0	0	0	0	0	...	0	0	
15118	0	0	0	1	0	0	0	0	0	0	...	0	0	
15119	0	0	0	1	0	0	0	0	0	0	...	0	0	
15120	0	1	0	0	0	0	0	0	0	0	...	0	0	

15120 rows × 41 columns

In [153]:
```
soil_columns['check'].sum(axis=0) # check if it adds up to 15120(nr of observations)
```

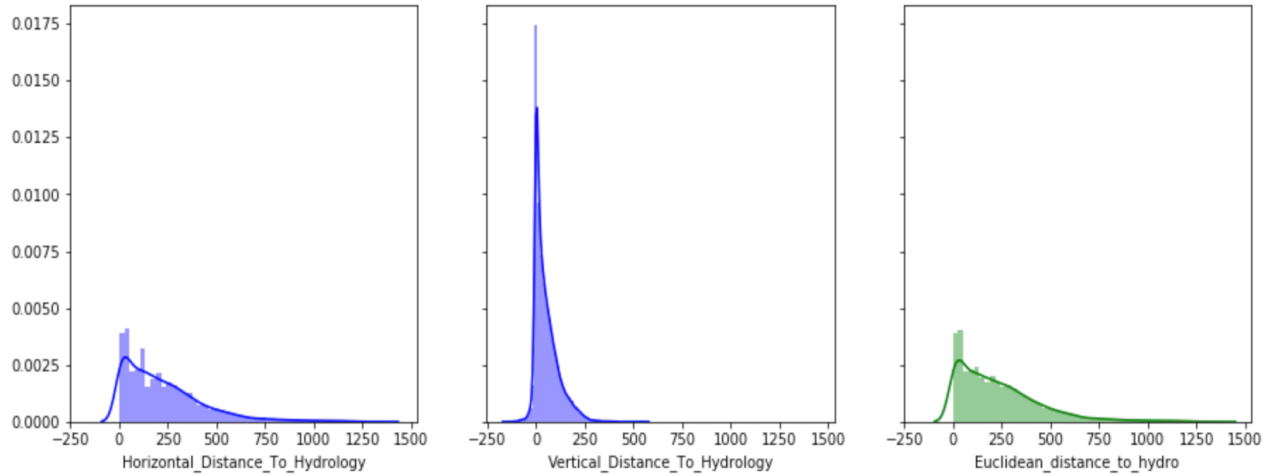
Out[153]: 15120

In [154]:
```
soil_columns['check'].unique() # yes there are only values as a 1, so 15120 rows with values 1 exist
```

Out[154]: array([1])

Data Exploration

- Since it is about trees I assume there is a relation with water. There are 2 features related to distanceand water:
  - Vertical_Distance_To_Hydrology
  - Horizontal_Distance_To_Hydrology



Interpretation

The first plot (horizontal distance to hydrology)
Vegetation seems to be more abundant near hydrology which makes sense.

The second plot (vertical distance) a huge amount of vegetation concentrated near 0, which means that much vegetation is at almost the same level of water.

When calculating the Euclidean distance to hydrology as a heuristic measure, we see that our third graph looks like the first, this is because the horizontal distance has a wider distribution compared to the vertical distance where almost all values are close to zero.

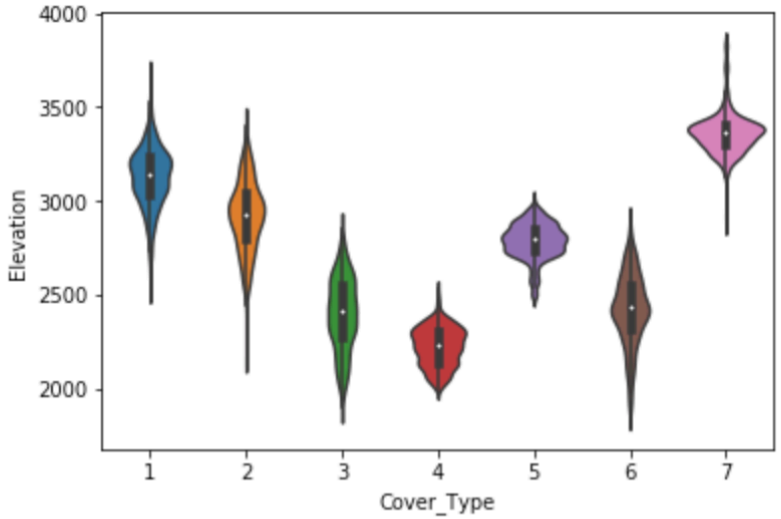
Data Visualization

- Elevation is interesting to see how it related to the target label

Check impact of elevation

```
In [163]: sns.violinplot(x=TARGET, y='Elevation', data=X)
```

```
Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26861fd0>
```



It looks like there is correlation between cover type and elevation.

Feature engineering(1)

- Based on distances some feature have been calculated to get more features

```python
def distances(df):
    cols = [
        'Horizontal_Distance_To_Roadways',
        'Horizontal_Distance_To_Fire_Points',
        'Horizontal_Distance_To_Hydrology',
    ]

    df['distance_mean'] = df[cols].mean(axis=1)
    df['distance_sum'] = df[cols].sum(axis=1)
    df['distance_road_fire'] = df[cols[:2]].mean(axis=1)
    df['distance_hydro_fire'] = df[cols[1:]].mean(axis=1)
    df['distance_road_hydro'] = df[[cols[0], cols[2]]].mean(axis=1)

    df['distance_sum_road_fire'] = df[cols[:2]].sum(axis=1)
    df['distance_sum_hydro_fire'] = df[cols[1:]].sum(axis=1)
    df['distance_sum_road_hydro'] = df[[cols[0], cols[2]]].sum(axis=1)

    df['distance_dif_road_fire'] = df[cols[0]] - df[cols[1]]
    df['distance_dif_hydro_road'] = df[cols[2]] - df[cols[0]]
    df['distance_dif_hydro_fire'] = df[cols[2]] - df[cols[1]]

    # Vertical distances measures
    colv = ['Elevation', 'Vertical_Distance_To_Hydrology']

    df['Vertical_dif'] = df[colv[0]] - df[colv[1]]
    df['Vertical_sum'] = df[colv].sum(axis=1)

    return df
```

Feature engineering(2)

- Soil types are categorized into 7 categories

Soil analysis

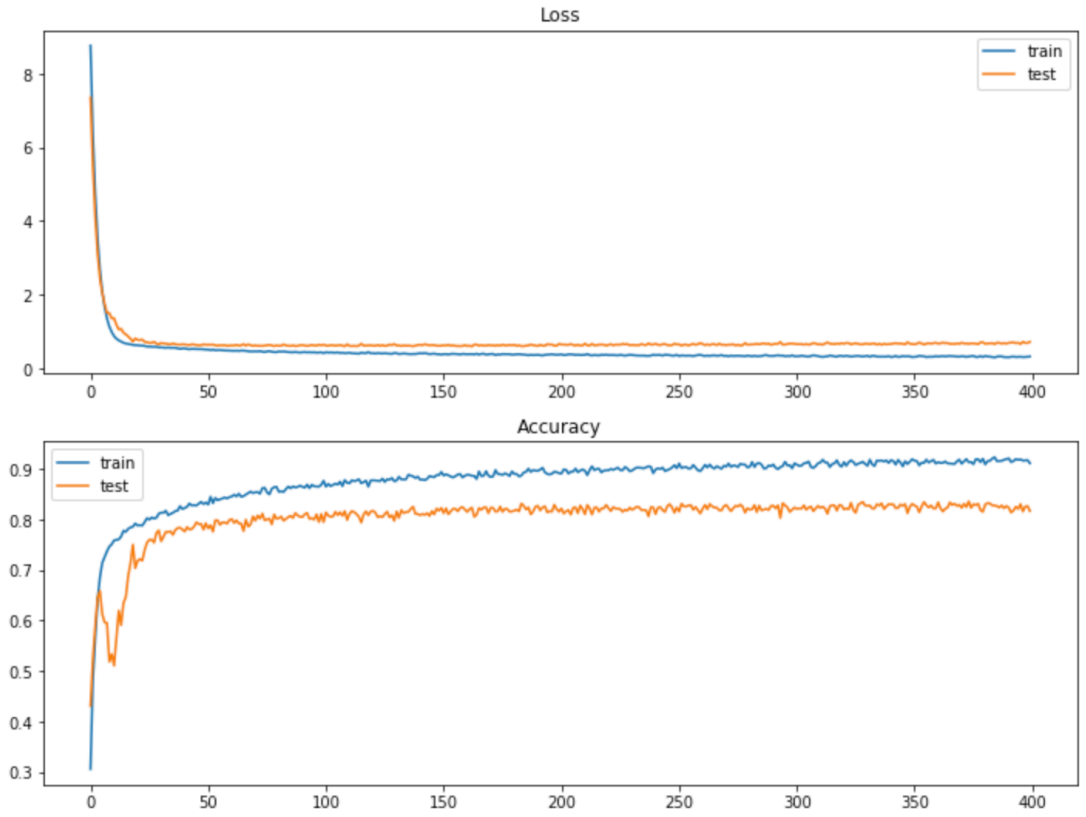
1=rubbly, 2=stony, 3=very stony, 4=extremely stony Above information is available in the covtype.info file from
https://archive.ics.uci.edu/ml/datasets/covertype

[28]:
```
# create a dict that map soil type with rockness
# 0=unknow 1=complex 2=rubbly, 3=stony,
# 4=very stony, 5=extremely stony 6=extremely bouldery
soils = [
    [7, 15, 8, 14, 16, 17,
     19, 20, 21, 23], #unknow and complex
    [3, 4, 5, 10, 11, 13],   # rubbly
    [6, 12],    # stony
    [2, 9, 18, 26],      # very stony
    [1, 24, 25, 27, 28, 29, 30,
     31, 32, 33, 34, 36, 37, 38,
     39, 40, 22, 35], # extremely stony and bouldery
]

soil_dict = dict()
for index, values in enumerate(soils):
    for v in values:
        soil_dict[v] = index

# if there is a 1 in a column then by multiplying with the index you get the Sol_Type number
def soil(df, soil_dict=soil_dict):
    df['Rocky'] =  sum(i * df['Soil_Type'+ str(i)] for i in range(1, 41))
    df['Rocky'] = df['Rocky'].map(soil_dict) #map if to 0 - 4
```

Model Performance Indicator

- I used a default from the sequence model for accuracy and loss which I could use to plot making the performance Visual. This gave me enough information on the performance

```
Train: 0.932, Test: 0.817
```



Machine learning algorithm used

- I used the randomForestClassifier with and without feature engineering.
- Without feature engineer gave an accuracy score of 0.84
- Using feature engineering the score went up to 0.88

Deep learning algorithm used

- I used Keras with sequential mode. No feature engineering was done.
- The accuracy score is 0.817 on the test dataset.

Model performance between different feature engineerings and models compared

Algorithm	Feature Engineering	Accuracy Score
RandomForestClassifier	No	0.84
RandomForestClassifier	Yes	0.88
Keras/Sequential Model	No	0.817

Conclusion: The neural network performs not too bad. I expected it to be more accurate but it is not. More investigation would be needed to see if it would make sense to use Neural Network for this use case to predict cover type.