# Pràctica 2 - Tipologia i cicle de vida de les dades

January 4, 2022

Òscar Busquets David Malvesí

---

En aquesta pràctica s'elabora un cas pràctic orientat a aprendre a identificar les dades rellevants per un projecte analític i usar les eines d'integració, neteja, validació i anàlisi de les mateixes.

## 1 Descripció del dataset. Perquè és important i quina pregunta/problema pretén respondre?

El dataset a treballar és part de la serie *Tabular Playground Series* i pertany al dataset penjat el mes de desembre de l'any 2021. El link a la competició de Kaggle el podem trobar aquí.

Aquest dataset prové del dataset de la competició Forest Cover Type Prediction i té com a objectiu la predicció de la coverta forestal a través de dades cartogràfiques. Les dades de l'estudi provenen de 4 areas diferents del *Roosvelt National Forest* del nord de Colorado i són parcel · les forestals que han patit una mínima interacció humana. És a dir, que l'ecosistema d'aquests boscos és producte de processos ecològics i no de gestió humana.

Entrant més en profunditat, cada punt del joc de dades representa una parcel · la de 30x30 metres i el tipus de coberta ha estat determinat pel *US Forest Service Region (USFUS) 2 Resource Information System Data*. Les variables independents han estat extretes del *US Geological Survey* i el USFS.

Cal tenir en compté, però, que el dataset que es farà servir és sintètic i ha estat generat fent servir CTGAN. Aquest és una col · lecció de generadors de dades sintètiques a través *Deep Learning* per a dades en format tabular. CTGAN aprén de les dades reals i genera dades sintètiques d'alta fidelitat.

### 1.1 Variables

**- Cover_Type** Aquesta variable és un valor enter i representa el tipus de coberta forestal. Es té 7 de diferents. 1. Avet (Spruce) 2. Pi contorta (Lodgepol Pine) 3. Pi ponderosa (Ponderosa Pine) 4. Salze (Aspen/Willow) 5. Pollancre (Aspen) 6. Avet de Douglas (Douglas-Fir) 7. Krummholz

**- Elevation** Elevació en metres.

**- Aspect** Aspecte en graus azimuth.

**- Slope** Pendent en graus.

**- Horizontal_Distance_To_Hydrology** Distància horitzontal en metres a la font d'aigua més propera.

**- Vertical_Distance_To_Hydrology** Distància vertical en metres a la font d'aigua més propera.

**- Horizontal_Distance_To_Roadways** Distància horitzontal en metres a la carretera més pròxima.

**- Hillshade_9am** Índex sobre l'ombra generada per la muntanya a les 9am, solstici d'estiu (index de 0 a 255).

**- Hillshade_Noon** Índex sobre l'ombra generada per la muntanya a les 12pm, solstici d'estiu (index de 0 a 255).

**- Hillshade_3pm** Índex sobre l'ombra generada per la muntanya a les 3pm, solstici d'estiu (index de 0 a 255).

**- Horizontal_Distance_To_Fire_Points** Distància horitzontal en metres fins als punts d'encesa forestal més proper.

**- Wilderness_Area** (4 binary columns, 0 = absence or 1 = presence) - Designació d'espai salvatge.

En aquest cas les 4 àreas naturals són: 1. Rawah 2. Neota 3. Comanche Peak. 4. Cache la Poudre.

**- Soil_Type** (40 binary columns, 0 = absence or 1 = presence) - Tipus de sol.

Els tipus de sól són: 1. Cathedral family - Rock outcrop complex, extremely stony. 2. Vanet - Ratake families complex, very stony. 3. Haploborolis - Rock outcrop complex, rubbly. 4. Ratake family - Rock outcrop complex, rubbly. 5. Vanet family - Rock outcrop complex complex, rubbly. 6. Vanet - Wetmore families - Rock outcrop complex, stony. 7. Gothic family. 8. Supervisor - Limber families complex. 9. Troutville family, very stony. 10. Bullwark - Catamount families - Rock outcrop complex, rubbly. 11. Bullwark - Catamount families - Rock land complex, rubbly. 12. Legault family - Rock land complex, stony. 13. Catamount family - Rock land - Bullwark family complex, rubbly. 14. Pachic Argiborolis - Aquolis complex. 15. unspecified in the USFS Soil and ELU Survey. 16. Cryaquolis - Cryoborolis complex. 17. Gateview family - Cryaquolis complex. 18. Rogert family, very stony. 19. Typic Cryaquolis - Borohemists complex. 20. Typic Cryaquepts - Typic Cryaquolls complex. 21. Typic Cryaquolls - Leighcan family, till substratum complex. 22. Leighcan family, till substratum, extremely bouldery. 23. Leighcan family, till substratum - Typic Cryaquolls complex. 24. Leighcan family, extremely stony. 25. Leighcan family, warm, extremely stony. 26. Granile - Catamount families complex, very stony. 27. Leighcan family, warm - Rock outcrop complex, extremely stony. 28. Leighcan family - Rock outcrop complex, extremely stony. 29. Como - Legault families complex, extremely stony. 30. Como family - Rock land - Legault family complex, extremely stony. 31. Leighcan - Catamount families complex, extremely stony. 32. Catamount family - Rock outcrop - Leighcan family complex, extremely stony. 33. Leighcan - Catamount families - Rock outcrop complex, extremely stony. 34. Cryorthents - Rock land complex, extremely stony. 35. Cryumbrepts - Rock outcrop - Cryaquepts complex. 36. Bross family - Rock land - Cryumbrepts complex, extremely stony. 37. Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony. 38. Leighcan - Moran families - Cryaquolls complex, extremely stony. 39. Moran family - Cryorthents - Leighcan family complex, extremely stony. 40. Moran family - Cryorthents - Rock land complex, extremely stony.

En resum, es treballa amb dataset sintètic provinent de dades reals on, a través de dades cartogràfiques, s'ha de predir el tipus de coberta forestal. L'objectiu serà predir el tipus de coberta a través de les dades de les que es dispossa. Per a fer això, serà molt important tant la comprensió de les dades amb les s'està treballant com un bon anàlisi i neteja de les dades.

## 2 Integració i selecció de les dades d'interés a analitzar.

L'objectiu de la integració es la de combinar diferents fonts de dades per tal de crear una estructura de dadaes coherent que contingui més informació.

Aquesta fusió es pot realitzar de dues maneres, de forma horitzontal -afegint columnes provinents d'altres fonts d'informació- o de forma vertical -afegint registres amb el mateix format-.

Cal tenir en compte que la competició ens proporciona dos jocs de dades el joc train i el joc test. El primer consta de 4 milions de files i el segon d'1 milió.

En el cas estudiat, es troba que no es pot fer cap integració horitzontal perquè la competició consta només d'un conjunt de dades amb un únic format. Cal tenir en compte, però, que aquestes dades ja són la fusió de multiples fonts d'informació com s'ha explicat a l'inici del document.

Pel que fa a la fusió vertical, es podria unir les dades de train amb test per a obtenir un dataset més gran. Tot i això es presenta un problema i és que el joc de dades test no conté la variable que es vol predir (*Cover_Type*), això és degut que l'objectiu de la competició és predit correctament el màxim de covertes forestals sobre el dataset test. Per aquest motiu no es tindrà en compte el dataset test, ja que no és útil per a l'anàlisi.

Així doncs, el que es farà és llegir el conjunt de dades i assegurar-se que no conté registres duplicats. Això es farà utilitzant la columna Id, la qual conté un número de registre únic per a cada observació.

[1]:
```
# Import the modules.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import chi2_contingency
from scipy.stats import f_oneway
from scipy.stats import shapiro

from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf
```

[2]:
```
 # Read the data.
original = pd.read_csv('data/train.csv')

# Check for duplicates in the Id columns.
if len(set(original['Id'])) == len(original['Id']):
```

```
    print('There are no duplicated Id\'s in the dataset.')
```

```
There are no duplicated Id's in the dataset.
```

Un cop comprobat que no es té cap registre duplicat es passa a la selecció de dades d'interés. En principi es necessita totes les dades del dataset, tot i això, es comprobarà quants registres es té per cada tipus de coberta forestal i prendre una decisió.

[3]:
```
# Check the number of observations by Cover_Type
original.groupby('Cover_Type').count()
```

[3]:

| Cover_Type | Id | Elevation | Aspect | Slope | \ |
|---|---|---|---|---|---|
| 1 | 1468136 | 1468136 | 1468136 | 1468136 | |
| 2 | 2262087 | 2262087 | 2262087 | 2262087 | |
| 3 | 195712 | 195712 | 195712 | 195712 | |
| 4 | 377 | 377 | 377 | 377 | |
| 5 | 1 | 1 | 1 | 1 | |
| 6 | 11426 | 11426 | 11426 | 11426 | |
| 7 | 62261 | 62261 | 62261 | 62261 | |

| Cover_Type | Horizontal_Distance_To_Hydrology | Vertical_Distance_To_Hydrology | \ |
|---|---|---|---|
| 1 | 1468136 | 1468136 | |
| 2 | 2262087 | 2262087 | |
| 3 | 195712 | 195712 | |
| 4 | 377 | 377 | |
| 5 | 1 | 1 | |
| 6 | 11426 | 11426 | |
| 7 | 62261 | 62261 | |

| Cover_Type | Horizontal_Distance_To_Roadways | Hillshade_9am | Hillshade_Noon | \ |
|---|---|---|---|---|
| 1 | 1468136 | 1468136 | 1468136 | |
| 2 | 2262087 | 2262087 | 2262087 | |
| 3 | 195712 | 195712 | 195712 | |
| 4 | 377 | 377 | 377 | |
| 5 | 1 | 1 | 1 | |
| 6 | 11426 | 11426 | 11426 | |
| 7 | 62261 | 62261 | 62261 | |

| Cover_Type | Hillshade_3pm | … | Soil_Type31 | Soil_Type32 | Soil_Type33 | \ |
|---|---|---|---|---|---|---|
| 1 | 1468136 | … | 1468136 | 1468136 | 1468136 | |
| 2 | 2262087 | … | 2262087 | 2262087 | 2262087 | |
| 3 | 195712 | … | 195712 | 195712 | 195712 | |
| 4 | 377 | … | 377 | 377 | 377 | |
| 5 | 1 | … | 1 | 1 | 1 | |

```
6                  11426  …        11426        11426        11426
7                  62261  …        62261        62261        62261

             Soil_Type34  Soil_Type35  Soil_Type36  Soil_Type37  Soil_Type38  \
Cover_Type
1                1468136      1468136      1468136      1468136      1468136
2                2262087      2262087      2262087      2262087      2262087
3                 195712       195712       195712       195712       195712
4                    377          377          377          377          377
5                      1            1            1            1            1
6                  11426        11426        11426        11426        11426
7                  62261        62261        62261        62261        62261

             Soil_Type39  Soil_Type40
Cover_Type
1                1468136      1468136
2                2262087      2262087
3                 195712       195712
4                    377          377
5                      1            1
6                  11426        11426
7                  62261        62261

[7 rows x 55 columns]
```

Clarament s'observa com hi ha dos grups infrarepresentats. El grup 4 (Salzes) amb 377 registres i el grup 5 (Pollancres) amb un sol registre.

Degut a la clarísima infrarepresentació del Pollancre s'eliminarà del dataset. De moment es mantindrà els salzes, tot i que durant l'anàlisi de les dades és possible que es decideixi suprimir-lo.

A més a més, també s'eliminarà la columna amb el número de registre ja que no té cap importància explicativa.

Seguidament es dividirà el dataset original entre les dades de train i test. Això és degut a que per aplicar un model supervisat es necessita mantenir una part del dataset sense estudiar. En aquest cas això serà un 20% de les dades originals.

[4]:
```
# Remove the Cover_Type 5 from the dataset.
original = original.loc[original['Cover_Type'] != 5, :]

# Remove the column Id.
original = original.loc[:, original.columns != 'Id']

# Split 80/20 to create train and test datasets.
train, test = train_test_split(original, stratify=original['Cover_Type'],␣
 ↪test_size=0.2)
```

Es crea una nova variable que representa la distància en linia recta fins

a la font d'aigua més propera. Aquesta variable s'anomenarà Diagonal_Distange_To_Hidrology i es calcularà de la següent manera: $Distance\_To\_Hydrology = \sqrt{Horizontal\_Distance\_To\_Hydrology^2 + Vertical\_Distance\_To\_Hydrology^2}$. En efecte, és el teorema de Pitàgoras.

[5]:
```
# Create the variable using the Pythagoras Theorem.
train['Distance_To_Hydrology'] = (train['Vertical_Distance_To_Hydrology'] ** 2
                                  + train['Horizontal_Distance_To_Hydrology']␣
 ↪** 2) ** (1/2)

test['Distance_To_Hydrology'] = (test['Vertical_Distance_To_Hydrology'] ** 2
                                 + test['Horizontal_Distance_To_Hydrology'] **␣
 ↪2) ** (1/2)

# Remove the Vertical_Distance_To_Hidrology and Horizontal_Distance_To_Hidrology
train = train.loc[:, ~train.columns.isin(['Horizontal_Distance_To_Hydrology',
                                          'Vertical_Distance_To_Hydrology',
                                          'Id'])]

test = test.loc[:, ~test.columns.isin(['Horizontal_Distance_To_Hydrology',
                                       'Vertical_Distance_To_Hydrology',
                                       'Id'])]
```

Es comproba si existeix algun registre on totes les variables Wilderness_Area singuin 0 o si pertanyen a més d'una Àrea Natural.

[6]:
```
# Generate a list with the Wilderness_Area columns columns.
wilderness_cols = [col for col in original.columns if 'Wilderness_Area' in col]

# Generate a pd.Serie with the sum of the 4 columns.
wilderness = original.loc[:, original.columns.isin(wilderness_cols)].sum(axis=1)
print('Hi ha {} registres que pertanyen a més d\'una àrea natual.'.
 ↪format(len(wilderness.loc[wilderness > 1])))
print('Hi ha {} registres que pertanyen a una àrea natual.'.
 ↪format(len(wilderness.loc[wilderness == 1])))
print('Hi ha {} registres que no pertanyen a cap àrea natual.'.
 ↪format(len(wilderness.loc[wilderness == 0])))
```

```
Hi ha 159868 registres que pertanyen a més d'una àrea natual.
Hi ha 3593099 registres que pertanyen a una àrea natual.
Hi ha 247032 registres que no pertanyen a cap àrea natual.
```

Si s'analitza visualment les diferents *Wilderness Areas* d'Estats Units al següent link s'observa com no hi ha cap tipus de *overlap* entre àreas, per tant, aquelles que pertanyen a més d'una àrea són incorrectes. Tot i així, com que les dades són sintètiques i no originals no s'eliminaran del conjunt de dades.

També es genera una nova columna **Wilderness_Area** que representarà el nombre d'àreas a les cuals pertany.

[7]:
```
# Create a column that represents if the register is into a Wilderness Area of␣
 ↪every observation.
train['Wilderness_Area'] = train.loc[:, train.columns.isin(wilderness_cols)].
 ↪sum(axis=1)
test['Wilderness_Area'] = test.loc[:, test.columns.isin(wilderness_cols)].
 ↪sum(axis=1)
```

Seguidament s'analitza per tipus de sól.

[8]:
```
# Generate a list with the Wilderness_Area columns columns.
soil_cols = [col for col in train.columns if 'Soil_Type' in col]

# Generate a pd.Serie with the sum of the 40 columns.
soil = train.loc[:, train.columns.isin(soil_cols)].sum(axis=1)
print('Hi ha {} registres que no contenen cap dels sóls.'.format(len(soil.
 ↪loc[soil == 0])))
print('Hi ha {} registres que contenen només un dels sóls.'.format(len(soil.
 ↪loc[soil == 1])))
print('Hi ha {} registres que contenen només més d\'un tipus de sól.'.
 ↪format(len(soil.loc[soil > 1])))
```

```
Hi ha 1325074 registres que no contenen cap dels sóls.
Hi ha 1249741 registres que contenen només un dels sóls.
Hi ha 625184 registres que contenen només més d'un tipus de sól.
```

En aquest s'observa registres de tots els tipus. A nivell teòric això no suposa cap problema ja que es poden donar totes les situacions. Es genera un nou atribut anomenat **Amount_Soil_Type** que serà la suma del nombre de sols que continguen. Aquest atribut clarament és interessant pel fet que representa els registres que contenen una barreja de sóls.

[9]:
```
# Create a column that represents the amount of Soil_Types of every observation.
train['Amount_Soil_Type'] = train.loc[:, train.columns.isin(soil_cols)].
 ↪sum(axis=1)
test['Amount_Soil_Type'] = test.loc[:, test.columns.isin(soil_cols)].sum(axis=1)
```

# 3 Neteja de les dades.

## 3.1 Les dades contenen zeros o elements buits? Com gestionaries aquests casos?

S'analitza la existència d'elements buits en el conjunt de dades.

[10]:
```
# Count the number of null variables.
count = 0
for idx, nulls in zip(train.isnull().sum().index, train.isnull().sum()):
    if nulls > 0:
        count += 1
        print('La variable {} conté {} valors nuls.'.format(idx, nulls))
```

```
if count == 0:
    print('No hi ha cap variable que contingui valors nuls.')
```

```
No hi ha cap variable que contingui valors nuls.
```

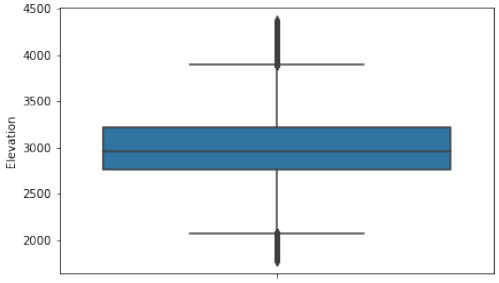
Es confirma que no hi ha cap valor buit en el joc de dades, tal i com ja estava confirmat al repositori de UCI Machine Learning. Si n'hi haguessim, es podria enfocar de la següent manera: 1. Identificar si són valors legitims com a *Amount_Type_Soil* o *Wilderness_Area*. Aquests valors no són per falta d'informació, sinò que senzillament no compleixen cap característica inicial. 2. Un cop identificats, si són legitims s'haurien d'etiquetar, en el present cas s'ha realitzat amb 0s. 3. Si no són legitims, o bé s'eliminen o bé s'inputa algun valor neutre com la mitjana/mediana/moda de les dades També es podria aplicar un algoritme de clustering i imputar-ne algun estadístic, etc.

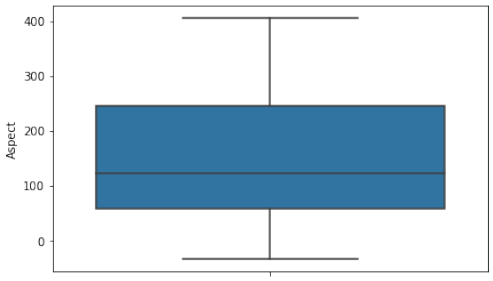
### 3.2 Identificació i tractament de valors extrems.

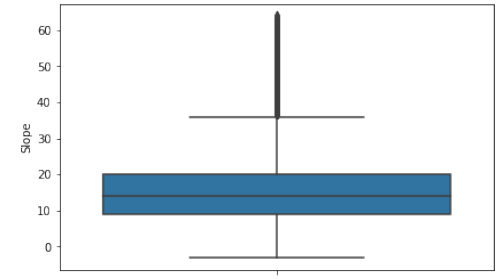
Els valors extrems o *outlier* són aquells valors molt allunyats de la distribució normal d'una variable. En el cas estudiat on s'està tractant amb boscos sense activitat humana de Colorado, només existeix un registre on la coverta forestal està coberta per pollancres. Sembla ser que aquest valor no era part de la població d'estudi, per exemple generada per l'activitat humana.

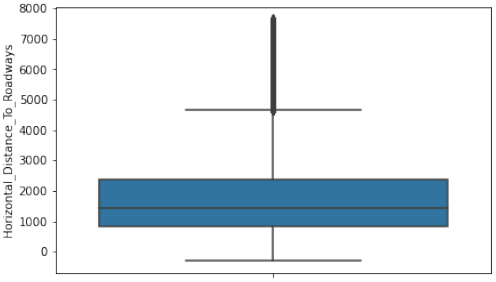
Pot ser que un valor estigui molt allunyat de la distribució d'una de les variables i tot i així tractar-se un *outlier* legitim.

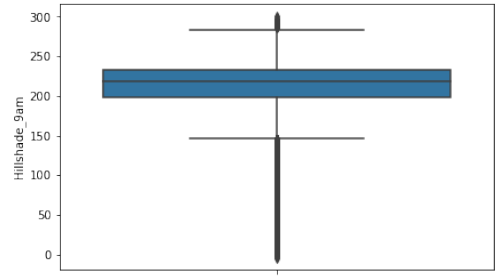
[11]:
```
to_plot =  ['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Roadways',
            'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
            'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area',
            'Distance_To_Hydrology', 'Amount_Soil_Type']

fig, axes = plt.subplots(int(round(len(to_plot)/2, 0)), 2, figsize=(15,30))

for row, feature in enumerate(to_plot):
    if row % 2 == 0:
        sns.boxplot(y=feature, data=train, ax=axes[int(row/2), 0])
    else:
        sns.boxplot(y=feature, data=train, ax=axes[int(row/2), 1])
plt.show()
```

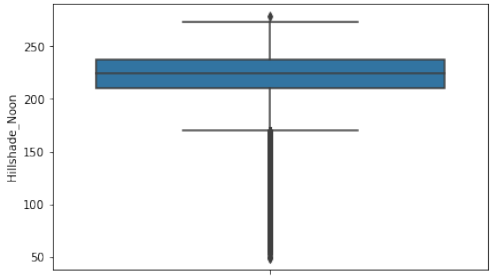

Elevation
4500
4000
3500
3000
2500
2000

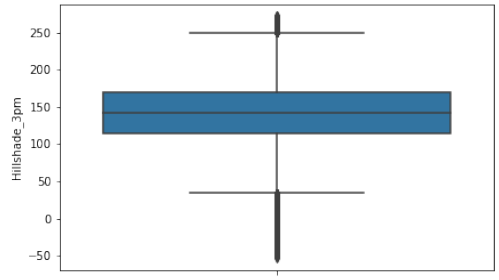

Aspect
400
300
200
100
0

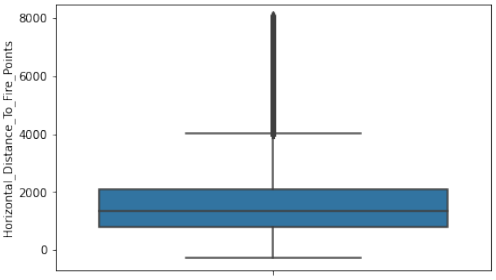

Slope
60
50
40
30
20
10
0


Horizontal_Distance_To_Roadways
8000
7000
6000
5000
4000
3000
2000
1000
0

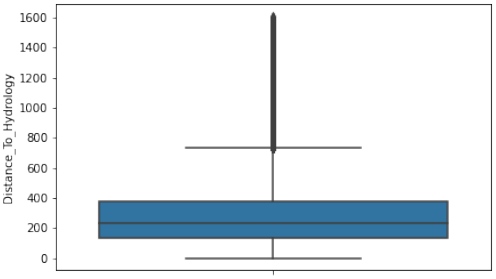

Hillshade_9am
300
250
200
150
100
50
0

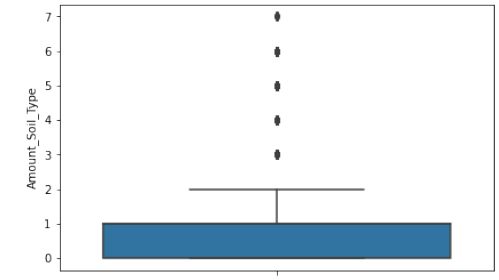

Hillshade_Noon
250
200
150
100
50

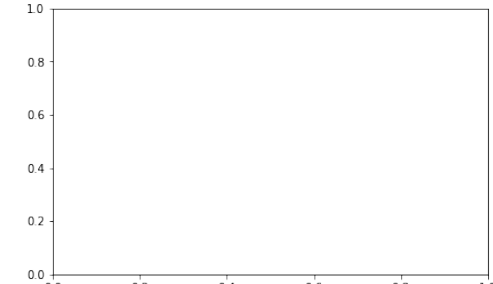

Hillshade_3pm
250
200
150
100
50
0
−50


Horizontal_Distance_To_Fire_Points
8000
6000
4000
2000
0


Wilderness_Area
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Distance_To_Hydrology
1600
1400
1200
1000
800
600
400
200
0


Amount_Soil_Type
7
6
5
4
3
2
1
0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Als gràfics anteriors s'observa multitud d'outliers, fora dels rangs del diagrama de caixes. Es mostra per variable quants outliers hi ha:

[12]:
```
# Find the outliers by variable in the dataset.
outliers = {}

for feature in to_plot:
    # Compute the average and standard deviation of the feature.
    average = train[feature].mean()
    std = train[feature].std()

    # Create a dictionary with the featre as a key and a list of outliers as␣
 ↪values.
    outliers[feature] = list(train.loc[(train[feature] > average + 3 * std) |
                                       (train[feature] < average - 3 * std)].
 ↪index)

# Generate a list with all the outliers.
outliers_total = []

for feature in outliers:
    outliers_total = outliers_total + outliers[feature]
    # Print the outliers per feature.
    print('{} has {} outliers, a {:2.2%} share of the data'.format(feature,
                                                                   ␣
 ↪len(outliers[feature]),
                                                                   ␣
 ↪len(outliers[feature])/len(train)))

print('\n')

# Print the share with the share of all the outliers at least in one feature.
print('In total there are {} outliers a {:2.2%} of the total'.
 ↪format(len(set(outliers_total)),
                                                                 ␣
 ↪len(set(outliers_total))/len(train)))
```

```
Elevation has 18180 outliers, a 0.57% share of the data
Aspect has 0 outliers, a 0.00% share of the data
Slope has 20247 outliers, a 0.63% share of the data
Horizontal_Distance_To_Roadways has 42786 outliers, a 1.34% share of the data
Hillshade_9am has 51688 outliers, a 1.62% share of the data
Hillshade_Noon has 44401 outliers, a 1.39% share of the data
Hillshade_3pm has 19904 outliers, a 0.62% share of the data
Horizontal_Distance_To_Fire_Points has 60926 outliers, a 1.90% share of the data
```

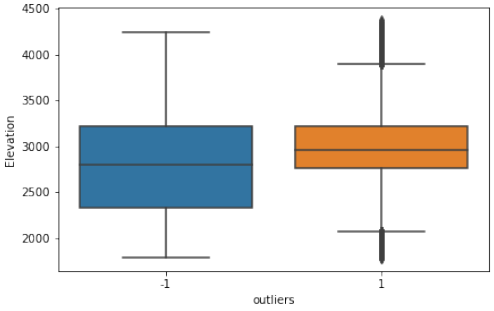
```
Wilderness_Area has 325620 outliers, a 10.18% share of the data
Distance_To_Hydrology has 69289 outliers, a 2.17% share of the data
Amount_Soil_Type has 18108 outliers, a 0.57% share of the data


In total there are 631017 outliers a 19.72% of the total
```

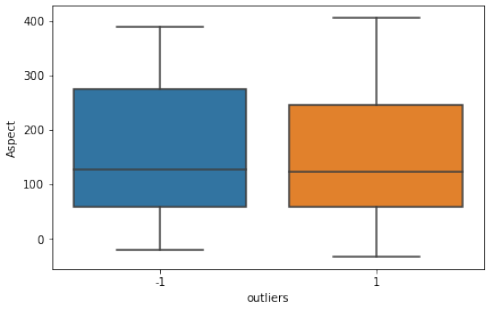
Més d'un 10% de les dades és considerat un *outlier* en algún dels atributs. No es pot considerar més de 300 mil punts de les dades com a valors extrems i eliminar-los. Degut a l'alta dimensionalitat de les dades tampoc es pot visualitzar-los de forma senzilla.

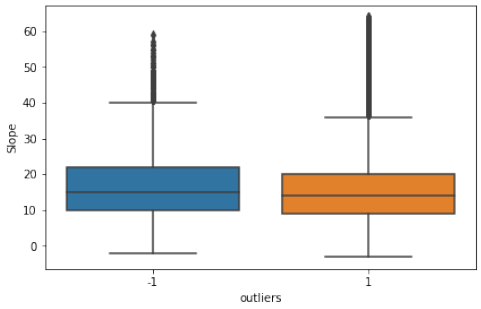
Llavors s'aplica l'algoritme Isolation Forest per a predir els outliers.

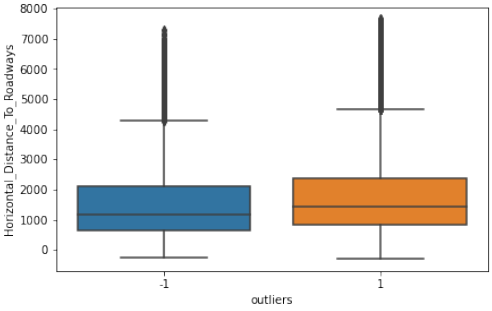
[13]:
```
# Create an Isolation Forest
iso = IsolationForest()

# Fit and predict using train data.
outliers_iso = iso.fit_predict(train)

# Print the amount of outliers.
print('The Isolation Forest detected {} outliers.'.format((outliers_iso == -1).
 ↪sum()))

# Create the column outliers
train['outliers'] = outliers_iso

# Gràfiquem les diferències entre grups.
fig, axes = plt.subplots(int(round(len(to_plot)/2, 0)), 2, figsize=(15,30))
for row, feature in enumerate(to_plot):
    if row % 2 == 0:
        sns.boxplot(y=feature, x='outliers', data=train, ax=axes[int(row/2), 0])
    else:
        sns.boxplot(y=feature, x='outliers', data=train, ax=axes[int(row/2), 1])
plt.show()

# Remove the column outliers.
train = train.loc[:, train.columns != 'outliers']
```

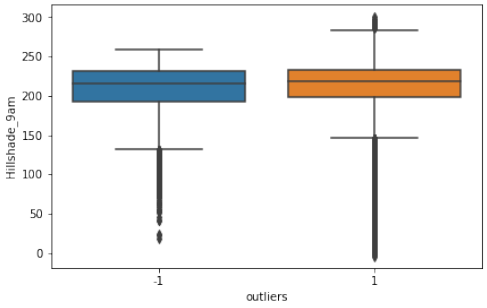
```
The Isolation Forest detected 6669 outliers.
```

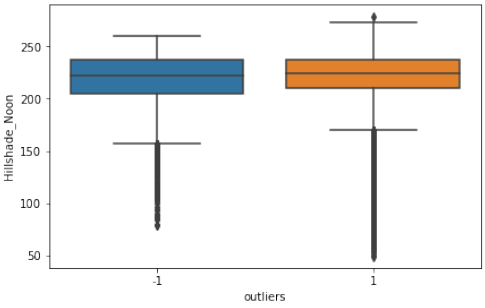

Elevation
outliers

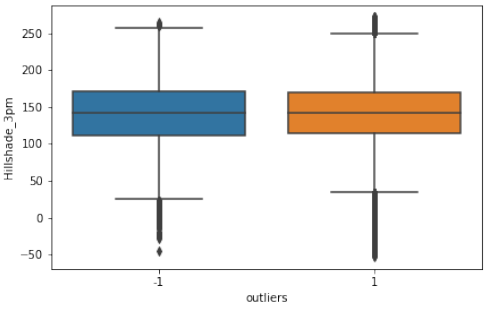

Aspect
outliers

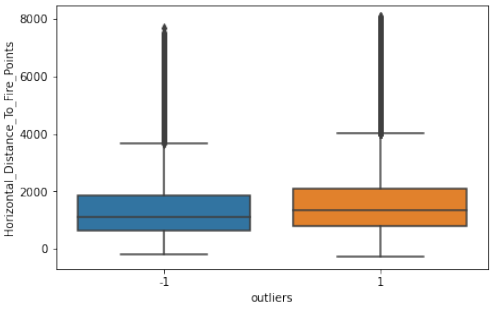

Slope
outliers

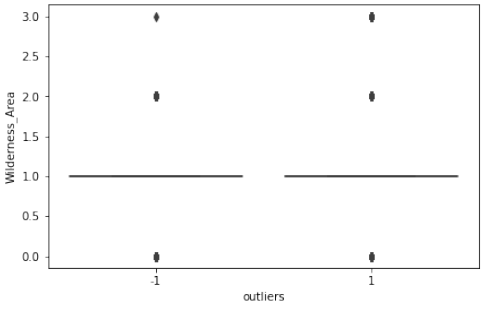

Horizontal_Distance_To_Roadways
outliers

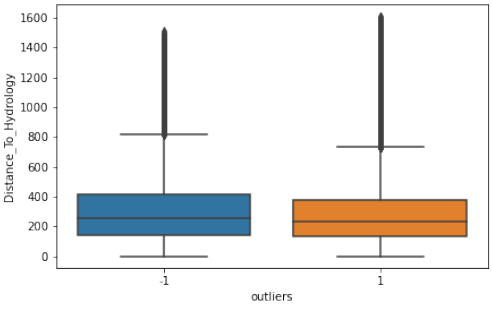

Hillshade_9am
outliers

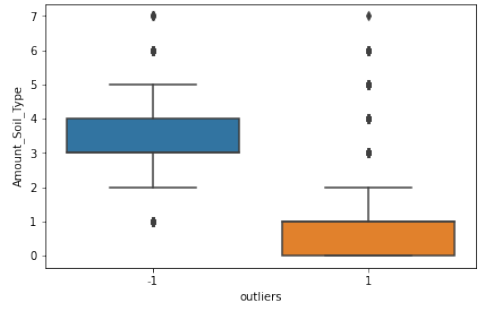

Hillshade_Noon
outliers

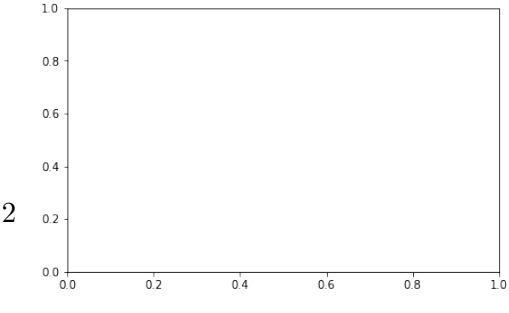

Hillshade_3pm
outliers


Horizontal_Distance_To_Fire_Points
outliers


Wilderness_Area
outliers


Distance_To_Hydrology
outliers


Amount_Soil_Type
outliers



Les desviacions estàndards de les variables estudiades són molt semblants. El cas de la variable **Elevation** i **Aspect** per outliers hi ha una desviació estàndard major. La gran diferència es troba a la variable creada **Amount_Soil_Type** on la mitjana de tipus de sóls en el cas dels outliers és 3.5 i als no outliers és 0. Degut a que no es desitja perdre els valors que tenen més d'un sol, es mantenen al dataset train i s'eliminen.

# 4 Anàlisi de les dades

## 4.1 Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar).

Es separa les dades originals segons el tipus de coberta forestal per a comparar-los.

[14]:
```
# Separate the data using the Cover_Type
spruce = train.loc[train['Cover_Type'] == 1]
lodgepol = train.loc[train['Cover_Type'] == 2]
ponderosa = train.loc[train['Cover_Type'] == 3]
aspen = train.loc[train['Cover_Type'] == 4]
douglas = train.loc[train['Cover_Type'] == 6]
krummholz = train.loc[train['Cover_Type'] == 7]
```

## 4.2 Comprovació de la normalitat i homogeneïtat de la variància.

S'estudia la normalitat de les variables quantitatives del dataset emprant la proba de Shapiro-Wilk. Es pretén acceptar la hipòtesis nul · la de normalitat de la població amb una confiança del 95%.

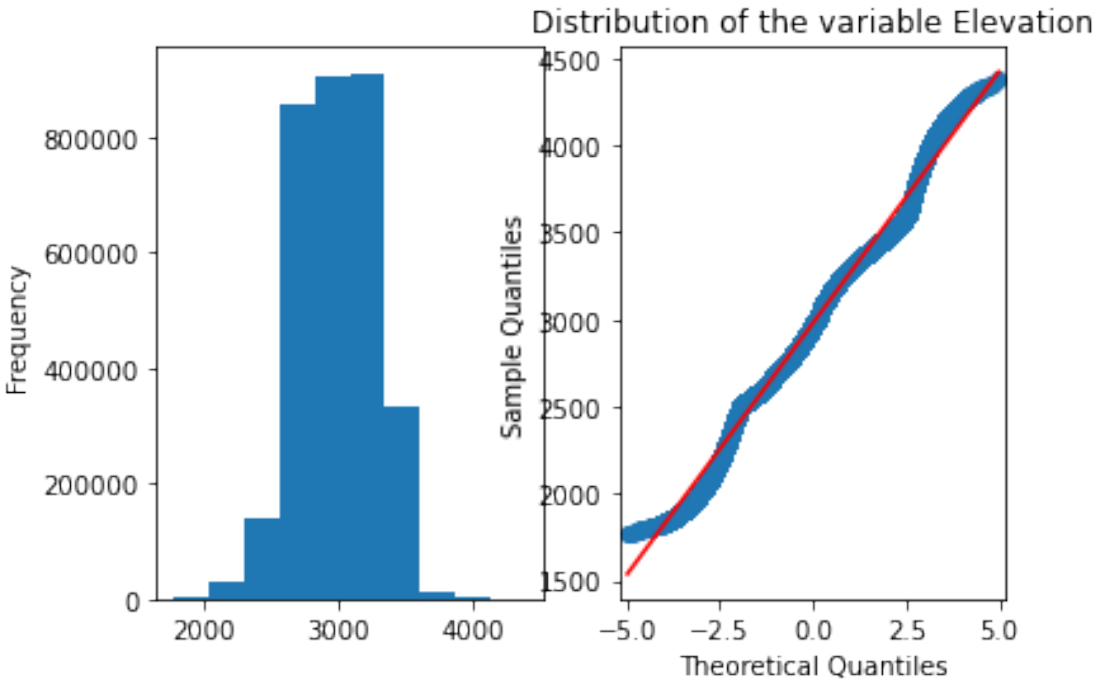
Si el p-value és inferior 0.05 es rebutja la hipòteis i s'assumeix que les dades no estan distribuides normalment. Si és major es conclou que no es pot rebutjar la hipòtesis nula de normalitat.

[15]:
```
normality = ['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Roadways',
             'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
             'Horizontal_Distance_To_Fire_Points', 'Distance_To_Hydrology']

# Perform the normality test for every continuous feature.
for feature in normality:
    # Print the results of the test.
    print('The test gives a p-value of {} for the feature {}'.
 ↪format(shapiro(train[feature])[1],
                                                                feature))
    # Plot the data to check the distribution visually.
    fig, ax = plt.subplots(1, 2)
    # Plot a histogram.
    train[feature].plot(kind='hist', ax=ax[0])
    # Plot a qq plot.
    sm.qqplot(train[feature], line='s', ax=ax[1])
    # Plot the title.
```

```
    plt.title('Distribution of the variable {}'.format(feature))
    # Show the subplots.
    plt.show()
```

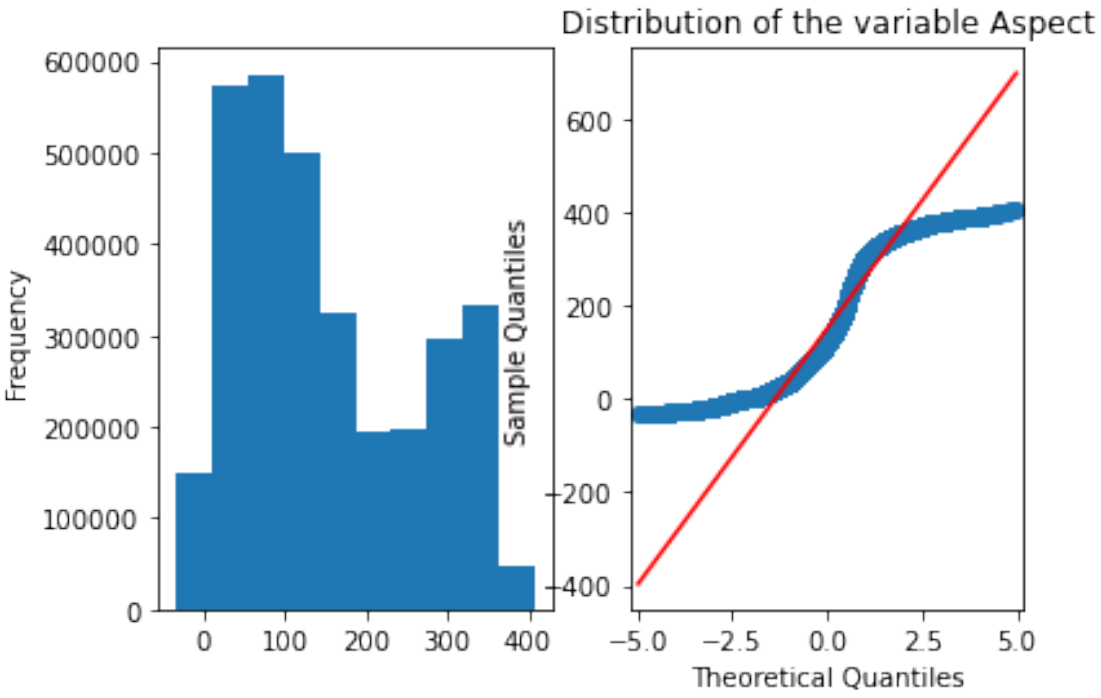
```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Elevation
```



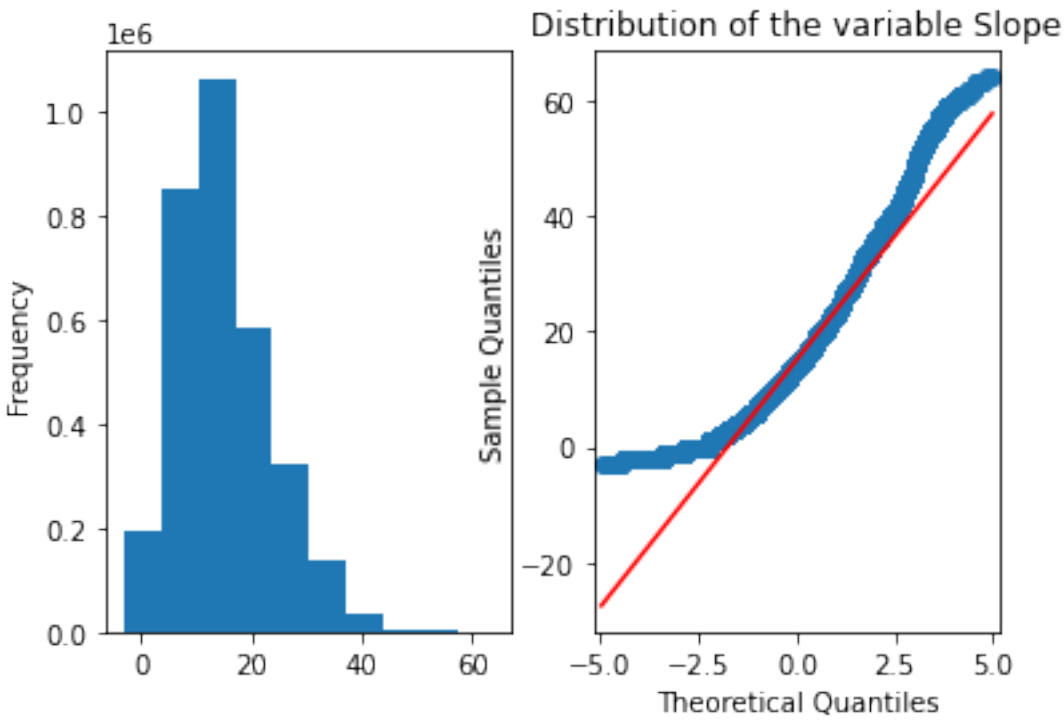

```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Aspect
```



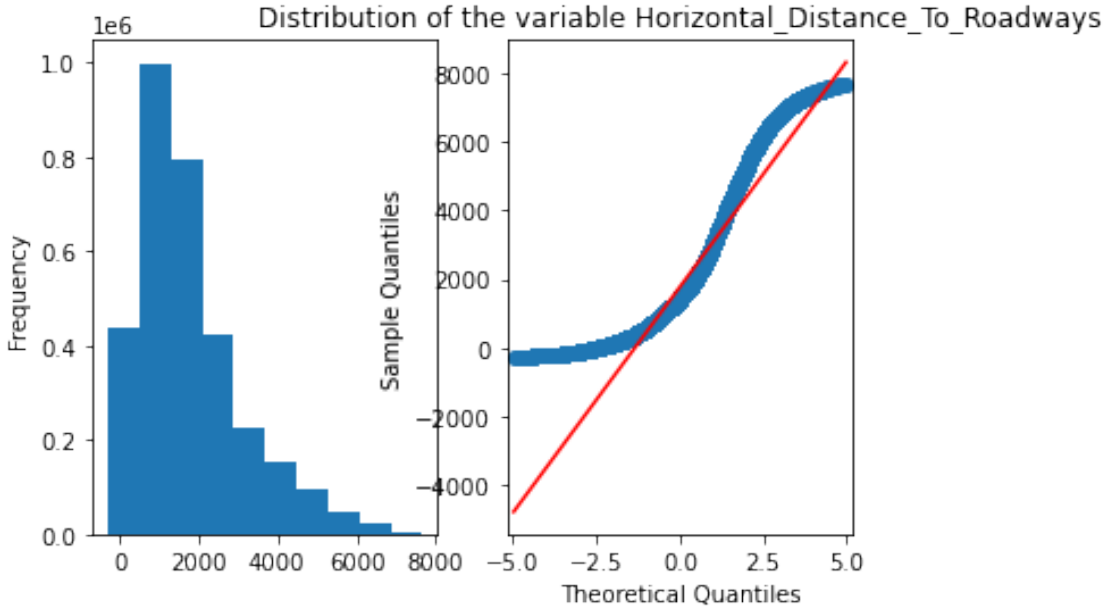

```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Slope
```



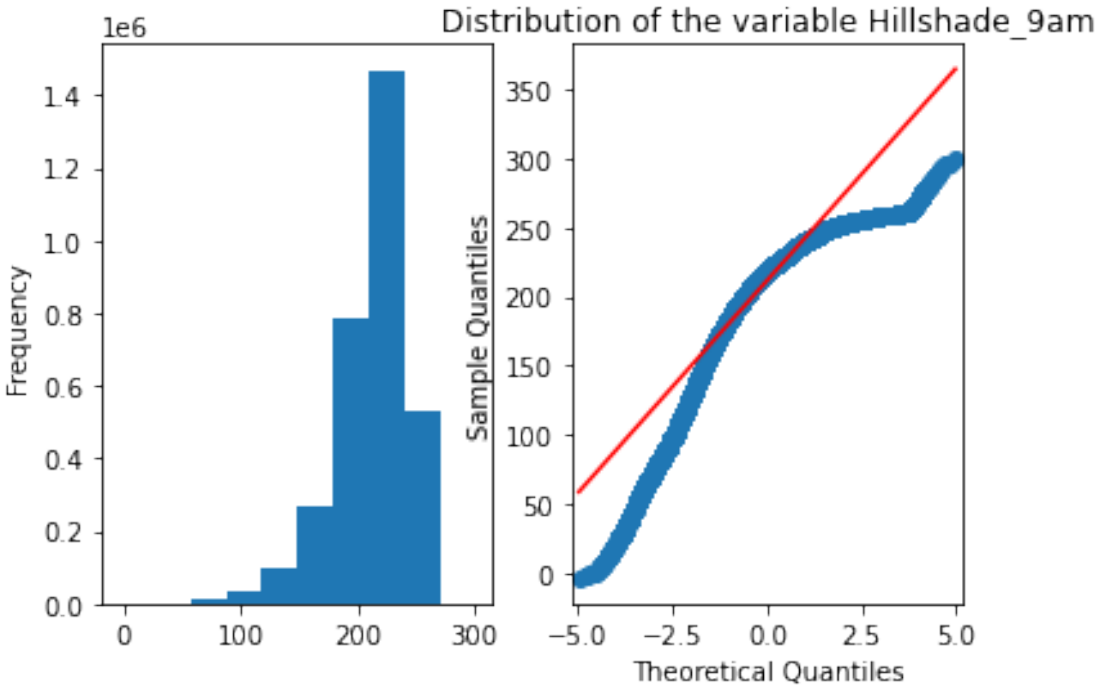

```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Horizontal_Distance_To_Roadways
```



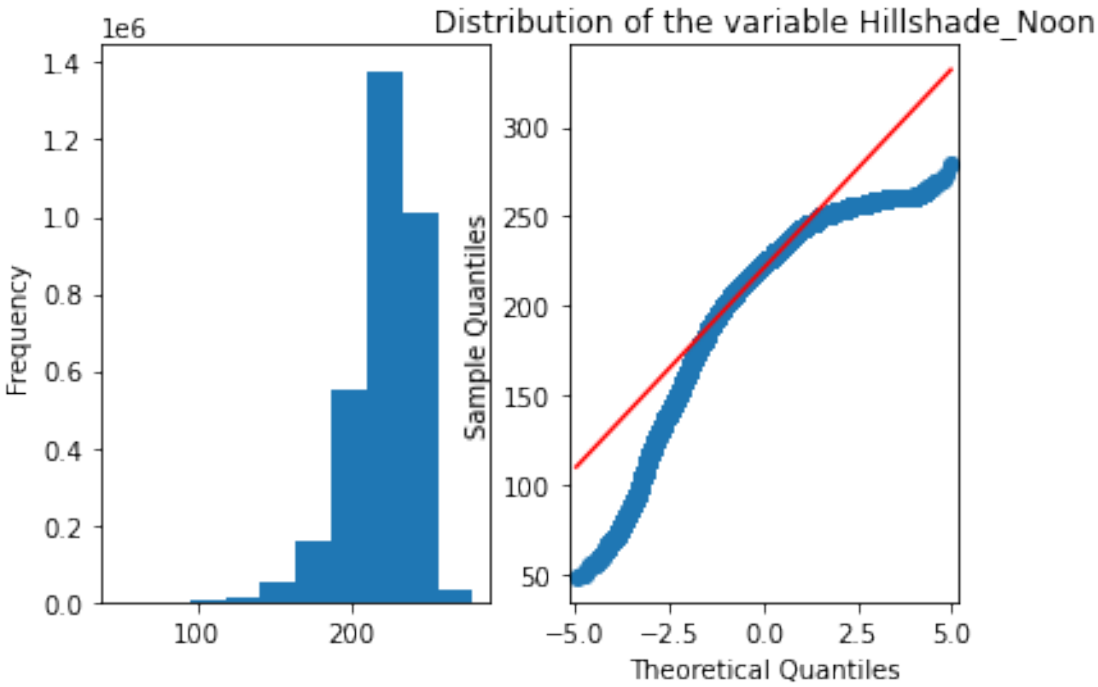
```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Hillshade_9am
```



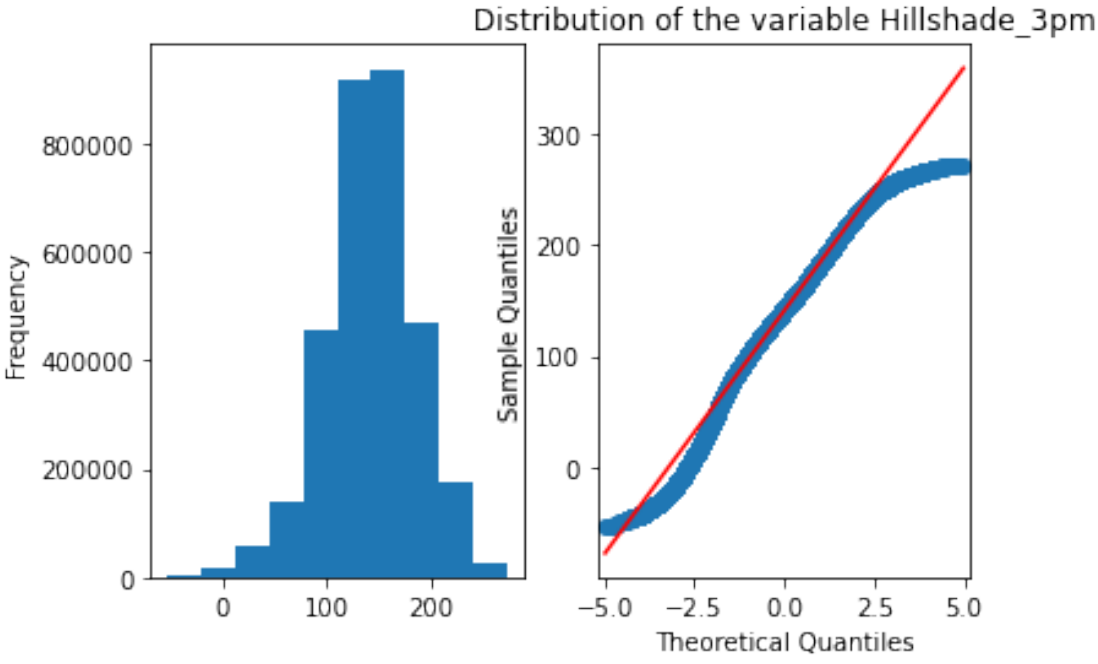

```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Hillshade_Noon
```



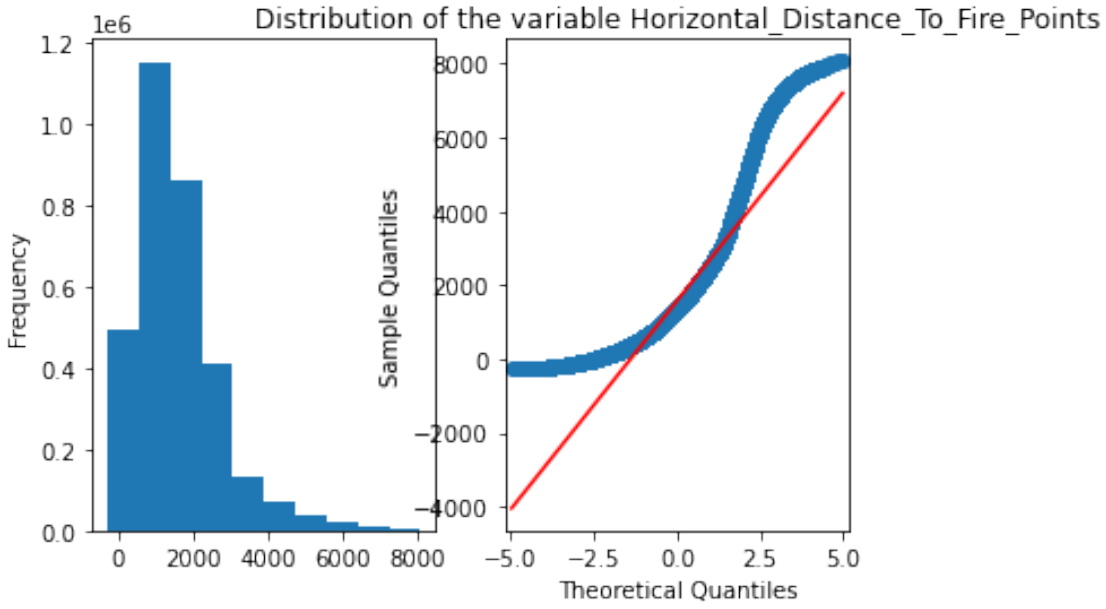

```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Hillshade_3pm
```



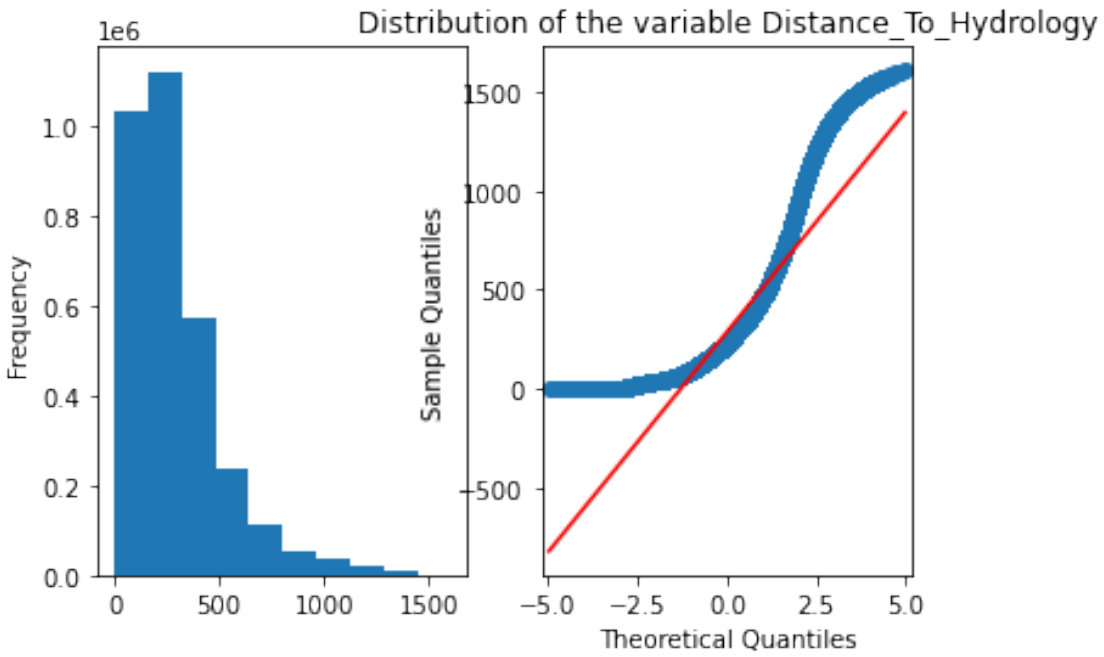
```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature
Horizontal_Distance_To_Fire_Points
```




```
/opt/anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```
The test gives a p-value of 0.0 for the feature Distance_To_Hydrology
```




Els tests en tots els casos rebutja les hipòtesis de normalitat en totes les variables, retornant un p-value de 0,00. Degut a que la mostra és molt gran, el test no funciona bé.

Si s'observa els histogrames, de forma subjectiva es podria determinar que la variable *Aspect* és l'única de totes elles que no segueix una normalitat. La resta manté unes campanes més o menys simètriques, però tenen certa tendència a la normalitat.

Es comproba l'homocedasticitat de les variables fent una regressió on Cover_Type és la variable dependent. Seguidament es procedeix amb el test Bresuch-Pagan i així controlar l'heterocedasticitat dels residus. Si el p-valor és més metit a 0.05 (confiança del 95%) es rebutja la hipòtesis nul · la d'homocedasticitat.

[16]:
```
# Perform an heteroskedasticity test for each variable.
for feature in normality:
    # Fit a linear regression with the Cover Type as dependant variable.
    fit = smf.ols('Cover_Type ~ {}'.format(feature), data=train).fit()
    # Perform the Breuch-Pagan test.
    het = sms.het_breuschpagan(fit.resid, fit.model.exog)
    if het[1] > 0.05:
        print('The p-value of {} for the Bresuch-Pagan test is {}'.
↪format(feature, het[1]))
        print('Hence we cannot reject the null hypothesis of homoscedasticity.
↪\n')
    else:
        print('The p-value of {} for the Bresuch-Pagan test is {}'.
↪format(feature, het[1]))
```

```
        print('We find evidence of heteroskedasticity.\n')
```

```
The p-value of Elevation for the Bresuch-Pagan test is 0.0
We find evidence of heteroskedasticity.

The p-value of Aspect for the Bresuch-Pagan test is 0.00015850257520374077
We find evidence of heteroskedasticity.

The p-value of Slope for the Bresuch-Pagan test is 5.110342862898386e-17
We find evidence of heteroskedasticity.

The p-value of Horizontal_Distance_To_Roadways for the Bresuch-Pagan test is 0.0
We find evidence of heteroskedasticity.

The p-value of Hillshade_9am for the Bresuch-Pagan test is
1.7824049601882288e-05
We find evidence of heteroskedasticity.

The p-value of Hillshade_Noon for the Bresuch-Pagan test is
0.00029183546903726286
We find evidence of heteroskedasticity.

The p-value of Hillshade_3pm for the Bresuch-Pagan test is 5.843389706503474e-17
We find evidence of heteroskedasticity.

The p-value of Horizontal_Distance_To_Fire_Points for the Bresuch-Pagan test is
1.5498658310882356e-87
We find evidence of heteroskedasticity.

The p-value of Distance_To_Hydrology for the Bresuch-Pagan test is
0.3943511771559436
Hence we cannot reject the null hypothesis of homoscedasticity.
```

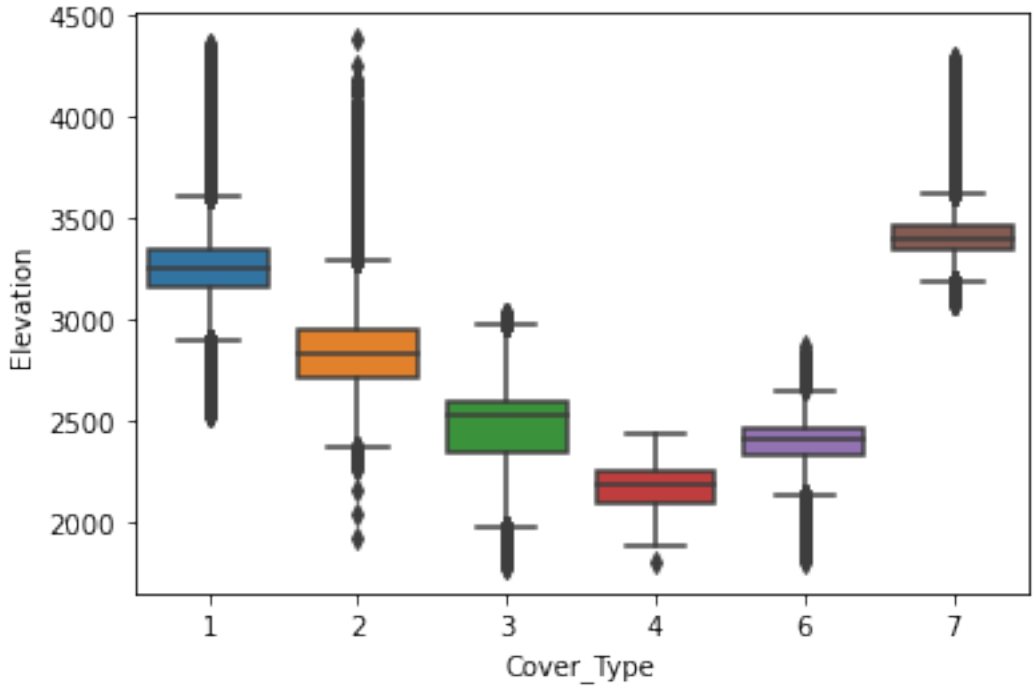
Hi ha una clara tendència a la heterocedasticitat en gairebé totes les variables, el que fa que no sigui tant homogènia i conclou variàncies diferents.

# 5 Aplicació de proves estadístiques per comparar els grups de dades. En funció de les dades i de l'objectiu de l'estudi, aplicar proves de contrast d'hipòtesis, correlacions, regressions, etc. Aplicar almenys tres mètodes d'anàlisis diferents.

## 5.1 Es l'elevació (*Elevation*) una bona variable explicativa de la Coverta Forestal (*Cover_Type*)?

S'empra l'ANOVA per a determinar si *Elevation* serà significativa en el model. Tanmateix s'assumeix la normalitat i independència de les variables tractades en els apartats anteriors.

[17]:
```
# Generate a dataframe with only the variables we want to test.
elevation = train.loc[:, ['Elevation', 'Cover_Type']]
# Plot a box plot.
sns.boxplot(x='Cover_Type', y='Elevation', data=elevation)
plt.show()

# Perform the one sided ANOVA test.
f_oneway(spruce['Elevation'], lodgepol['Elevation'], ponderosa['Elevation'],
         aspen['Elevation'], douglas['Elevation'], krummholz['Elevation'])
```




[17]:
```
F_onewayResult(statistic=1518994.8416567324, pvalue=0.0)
```

Amb un p-value més petit de 0.05 i l'estadístic F molt gran es rebutja la hipòtesis nul · la de no diferència entre les mitjanes dels grups i concloure que Elevation és una bona variable explicativa del model.

## 6 La coberta forestal depén de l'Àrea Natural?

En aquest cast s'aplicarà l'estadistic Chi-Quadrat on la hipotesis nul · la és la independència de les dues variables categòriques.

[18]:
```
# Select the Wilderness_Area columns.
wilderness = ['Wilderness_Area1', 'Wilderness_Area2',
              'Wilderness_Area3', 'Wilderness_Area4']
```

```
# Create a column that codifies the belonging to a Wilderness_Area.
train['Wilderness_Code'] = train[wilderness].apply(lambda row: '_'.join(row.
 ↪values.astype(str)), axis=1)

# Create a dictionary to replace the code by a name we understand.
area_code = {'0_0_0_0': 'No Area',
             '0_0_0_1': 'Area 4',
             '0_0_1_0': 'Area 3',
             '0_0_1_1': 'Area 3+4',
             '0_1_0_0': 'Area 1',
             '0_1_0_1': 'Area 1+4',
             '0_1_1_0': 'Area 2+3',
             '1_0_0_0': 'Area 1',
             '1_0_1_0': 'Area 1+3',
             '1_1_0_0': 'Area 1+2',
             '1_1_1_0': 'Area 1+2+3'}

# Make the replacement.
train['Wilderness_Code'] = train['Wilderness_Code'].map(area_code)

# Create a contingency table between the two variables of interest.
contingency_table = pd.crosstab(train['Cover_Type'],
                                train['Wilderness_Code'],
                                margins=False)

# Print the contingency table.
print(contingency_table)

# Perform the Chi Squared test on the contingency table.
chi_test = chi2_contingency(contingency_table)

# Print the p-value of the Chi Squared.
print('Thi Chi squared test returns a p-value of {}'.format(chi_test[1]))

# Remove the variable just created.
train = train.loc[:, train.columns != 'Wilderness_Code']
```

```
Wilderness_Code  Area 1  Area 1+2  Area 1+2+3  Area 1+3  Area 1+4  Area 2+3  \
Cover_Type
1                356606      9683          67     23654         0     20157
2                460808     11095          52     29354        26     30273
3                    14         1           0        13       647      1012
4                     0         0           0         0         0         0
6                     0         0           0         0        89        17
7                  2584       121           1       662         0      1186
```

```
Wilderness_Code   Area 3  Area 3+4  Area 4  No Area
Cover_Type
1                 644564         0       0   119778
2                1198878         3    2117    77063
3                  95013        19   59843        7
4                      0         0     302        0
6                   2170        11    6850        4
7                  44630         0       0      625
Thi Chi squared test returns a p-value of 0.0
```

Amb p-valor més petit que 0.05 es rebutja la hipòtesis nul · la de independència en les variables i, per tant, es pot assumir que la coberta forestal i l'àrea natural són variables dependents, on es podria prescindir d'una d'elles.

### 6.1 Es pot predir la coberta forestal a partir de les dades cartogràfiques?

Finalment es respon la pregunta inicial de l'anàlisis. Es pot predir la coberta forestal? Per fer-ho s'utilitza un *Decission Tree Classifier*, l'elecció d'aquest algoritme serà l'avantatge de facilitat en la interpretació.

L'algoritme ha estat entrenat fent servir cross validation per a trobar la profunditat òptima. Seguidament s'ha entrenat l'arbre amb la profunditat que s'ha obtingut i testejat sobre els valors de test.

[23]:
```
# Split the data between features and values to predict.
train_X = train.loc[:, train.columns != 'Cover_Type']
train_y = train.loc[:, train.columns == 'Cover_Type']
test_X = test.loc[:, test.columns != 'Cover_Type']
test_y = test.loc[:, test.columns == 'Cover_Type']

# Use cv to test multiple maximum depths of the tree.
# We keep this part of the code comented because it takes an hour to be runned.
#cv_scores = {}
#
#for depth in range(30, 55, 5):
#    clf = DecisionTreeClassifier(max_depth=depth, min_samples_leaf=50,␣
 ↪random_state=0)
#    scores = cross_val_score(clf, train_X, train_y, cv=3)
#    cv_scores[depth] = scores.mean()

# From the hyperparameters tested we get that at 45 levels of depth the model
# converges and achieves a mean accuracy of 94.62962332700039%.

# Fit the decision tree with a max depth of 45.
clf = DecisionTreeClassifier(max_depth=45, min_samples_leaf=50, random_state=0)
clf = clf.fit(train_X, train_y)
print('The accuracy of the tree in the test set is {:2.2%}'.format(clf.
 ↪score(test_X, test_y)))
# Predict the test_X.
```

```
predicted = clf.predict(test_X)
# Print the confussion matrix and the classification report.
print('Confussion Matrix:\n',
      confusion_matrix(test_y, predicted,
                       labels=list(set(test_y.values.reshape(1,-1)[0]))))

print('\nClassification report:\n',
      classification_report(test_y, predicted,
                            labels=list(set(test_y.values.reshape(1,-1)[0]))))

# Print the features and the importance of each attribute.
for feature, importance in zip(clf.feature_names_in_, clf.feature_importances_):
    print('La importància de {} al model és {:2.2%}'.format(feature,␣
 ↪importance))
```

```
The accuracy of the tree in the test set is 94.70%
Confussion Matrix:
 [[280339  11002      0      0      0   2286]
 [ 14896 433402   3869      0    173     78]
 [     0   4804  33779      8    552      0]
 [     0      0     64     11      0      0]
 [     0    326    855      1   1103      0]
 [  3404     92      0      0      0   8956]]

Classification report:
               precision    recall  f1-score   support

           1       0.94      0.95      0.95    293627
           2       0.96      0.96      0.96    452418
           3       0.88      0.86      0.87     39143
           4       0.55      0.15      0.23        75
           6       0.60      0.48      0.54      2285
           7       0.79      0.72      0.75     12452

    accuracy                           0.95    800000
   macro avg       0.79      0.69      0.72    800000
weighted avg       0.95      0.95      0.95    800000

La importància de Elevation al model és 79.05%
La importància de Aspect al model és 0.09%
La importància de Slope al model és 0.05%
La importància de Horizontal_Distance_To_Roadways al model és 4.73%
La importància de Hillshade_9am al model és 0.08%
La importància de Hillshade_Noon al model és 0.12%
La importància de Hillshade_3pm al model és 0.07%
La importància de Horizontal_Distance_To_Fire_Points al model és 4.06%
La importància de Wilderness_Area1 al model és 1.41%
```

```
La importància de Wilderness_Area2 al model és 0.04%
La importància de Wilderness_Area3 al model és 2.23%
La importància de Wilderness_Area4 al model és 0.33%
La importància de Soil_Type1 al model és 0.05%
La importància de Soil_Type2 al model és 0.26%
La importància de Soil_Type3 al model és 0.11%
La importància de Soil_Type4 al model és 0.27%
La importància de Soil_Type5 al model és 0.05%
La importància de Soil_Type6 al model és 0.01%
La importància de Soil_Type7 al model és 0.00%
La importància de Soil_Type8 al model és 0.00%
La importància de Soil_Type9 al model és 0.00%
La importància de Soil_Type10 al model és 0.11%
La importància de Soil_Type11 al model és 0.01%
La importància de Soil_Type12 al model és 0.00%
La importància de Soil_Type13 al model és 0.00%
La importància de Soil_Type14 al model és 0.00%
La importància de Soil_Type15 al model és 0.00%
La importància de Soil_Type16 al model és 0.00%
La importància de Soil_Type17 al model és 0.00%
La importància de Soil_Type18 al model és 0.00%
La importància de Soil_Type19 al model és 0.00%
La importància de Soil_Type20 al model és 0.00%
La importància de Soil_Type21 al model és 0.00%
La importància de Soil_Type22 al model és 0.11%
La importància de Soil_Type23 al model és 0.04%
La importància de Soil_Type24 al model és 0.00%
La importància de Soil_Type25 al model és 0.00%
La importància de Soil_Type26 al model és 0.00%
La importància de Soil_Type27 al model és 0.00%
La importància de Soil_Type28 al model és 0.00%
La importància de Soil_Type29 al model és 0.00%
La importància de Soil_Type30 al model és 0.00%
La importància de Soil_Type31 al model és 0.01%
La importància de Soil_Type32 al model és 0.02%
La importància de Soil_Type33 al model és 0.02%
La importància de Soil_Type34 al model és 0.00%
La importància de Soil_Type35 al model és 0.09%
La importància de Soil_Type36 al model és 0.04%
La importància de Soil_Type37 al model és 0.07%
La importància de Soil_Type38 al model és 0.40%
La importància de Soil_Type39 al model és 0.45%
La importància de Soil_Type40 al model és 0.24%
La importància de Distance_To_Hydrology al model és 1.87%
La importància de Wilderness_Area al model és 0.00%
La importància de Amount_Soil_Type al model és 3.48%
```

L'exactitud del model és molt adequada amb el 94,72%. Remarcar la variable explicativa

més important que és l'*Elevation* seguida de *Horizontal_Distance_To_Roadways* i *Horizontal_Distance_To_Fire_Point*. Aquestes variables explicarien la vegetació d'algunes de les parcel · les per proximitat a d'altres. L'altre variable explicativa del model és la variable *Amount_Soil_Type* que explica els diferents tipus de sols que conté la parcel · la.

Analitzant més en detall s'observa que el fet que nombre d'observacions d'entrenament sigui tan poc homogènia, provoca que l'algoritme tingui més facilitat per aprendre quines característiques separen millor cada categoria i en conseqüència no millora la precisió del model en les categories menys represetnades. Per exemple la precisió de les covertes forestals 1 i 2 són del 94% i 95% correctament classificades. Tanmateix, la coberta 6, la més infrarepresentada, té una precisió de només un 63%.

Per tant, seria interessant probar que succeeix si es millora el balanç del conjunt de dades o probar diferents algoritmes més potents que l'arbre de classificació.

# 7 Conclusions

El present estudi ha treballat el dataset del mes de desembre del 2021 de la sèrie Tabular Playground Series on l'objectiu és la predicció de la coverta forestal. El principal objectiu de l'estudi ha sigut possar a prova el dataset i comprobar l'exactitud d'un model elaborat, en aquest cas un arbre de classificació.

S'ha començat treballant el datasset fent una neteja i un anàlisi de les variables. El conjunt de dades no contenia dades buides però sí valors extrems. Tanmateix s'ha estudiat la normalitat, homocedasticitat i independència de les variables.

Finalment, s'ha posat a prova elaborant un model supervissat. La pregunta inicial de si és possible predir el tipus de coberta forestal d'una parcel · la de 30x30m a partir de les dades cartografiques, ha quedat resolta amb un model d'arbre de classificació que aconsegueix una exactitud *out-of-sample* del 94.72%, és a dir, que es prediu correctament la gran majoria de les dades.

Caldria però, esmentar que les dades que es dispossaven no eren homogènies, ja que el tipus de coverta forestal no estava representada en parts iguals, sinò que la majoria eren de dos típus que acaba distorsionant les que estan infrarepresentades.

**Descàrrega dataset**

Finalment, es descarrega el dataset després de l'anàlisis.

[25]:
```
train.to_csv('final_data.csv')
```

| Contribucions | Firma |
|---|---|
| Investigació prèvia | Òscar, David |
| Redacció de les respostes | Òscar, David |
| Desenvolupament codi | Òscar, David |