Forest Cover Type

Karolína Solanská, semestrální práce MI-IKM

Data

Jako dataset jsem si vybrala https://www.kaggle.com/uciml/forest-cover-type-dataset. Je to volně přístupný dataset a obsahuje pozorování stromů ze čtyř oblastí Roosevelt National Forest v Coloradu. Informace obsažené v datasetu jsou: druh stromu, pokrytí stínu, vzdálenost k nejbližším významným bodům (cesty), typ půdy a místní topografie. Tato data byla původně součástí UCI Machine Learning Repository, a originální zdroj je k nalezení zde: https://archive.ics.uci.edu/ml/datasets/Covertype.
Na základě těchto dat se dá postavit predikční model, který bude předpovídat, které typy stromů budou růst v jakém terénu a jak daleko například od jiných významných bodů v krajině.
Tvrdá fakta (statistiky):

- **Instancí**: 581012
- **Features**:
  - Elevation (nadmořská výška)
    - Size: [581012 1]
    - Type: 'double', Min: 1859, Median: 2996, Max: 3858, NumMissing: 0
  - Aspect (směr nejstrmějšího svahu) - ve stupních azimutu
    - Size: [581012 1]
    - Type: 'double', Min: 0, Median: 127, Max: 360, NumMissing: 0
  - Slope (sklon svahu) - ve stupních
    - Size: [581012 1]
    - Type: 'double', Min: 0, Median: 13, Max: 66, NumMissing: 0
  - Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Roadways, Horizontal_Distance_To_Fire_Points
  - Hillshade_9am, Hillshade_Noon, Hillshade_3pm
  - Wilderness_Area1, Wilderness_Area2, Wilderness_Area3, Wilderness_Area4
    - Těmito hodnotami jsou označeny čtyři oblasti v rámci parku, hodnota je 0, nebo 1, podle toho, zda toto měření proběhlo v této oblasti
  - Soil_Type1 - Soil_Type40
    - Takto jsou evidovány typy půdy 1 až 40, kterým je přiřazena legenda, hodnota 0, nebo 1, podle toho, jestli to byl tento typ půdy, nebo ne
  - Cover_Type
    - hodnoty 1 až 7 označují jednotlivé druhy stromů rostoucí v Roosevelt National Forest
    - 1 - Spruce/Fir (Smrk/Jedle), 2 - Lodgepole Pine (Borovice pokroucená), 3 - Ponderosa Pine (Borovice těžká), 4 - Cottonwood/Willow (Topol/Vrba), 5 - Aspen (Topol), 6 - Douglas-fir (Douglaska tisolistá), 7 - Krummholz (zdeformované dřeviny)
    - V přiložené tabulce vidíme shrnuté, který druh dřevin se v parku vyskytuje nejčastěji - borovice pokroucená. Instancí je víc, než dat,

vzhledem k tomu, že na jednom místě s určitými podmínkami roste obvykle více různých typů stromů

Cover Type	
1	211,840
2	566,602
3	107,262
4	10,988
5	47,465
6	104,202
7	143,570

Předzpracování dat

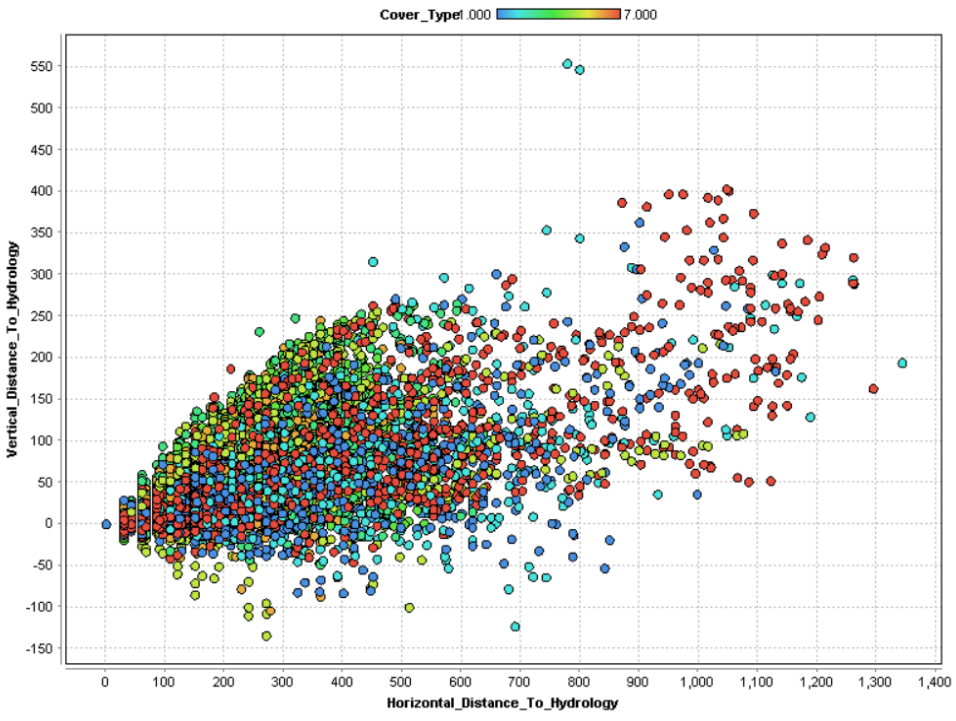
Vzhledem k tomu, že data jsou čistá a plná, nebylo třeba řešit chybějící hodnoty a pro testování mi zůstaly všechny hodnoty. To skýtá výhodu velkého množství dat. Po analýze složení dat vyšlo najevo, že není nutné je formalizovat.
Nekonzistenci dat na takto rozsáhlém datasetu mohu zjišťovat jen těžko, vzhledem k tomu, že nevím, co by měly být nekonzistentní hodnoty. Nicméně z průměrných hodnot, maxima a minima nebo mediánu se dá určit, že hodnoty nikde nejdou do extrémů, které by byly podezřelé.

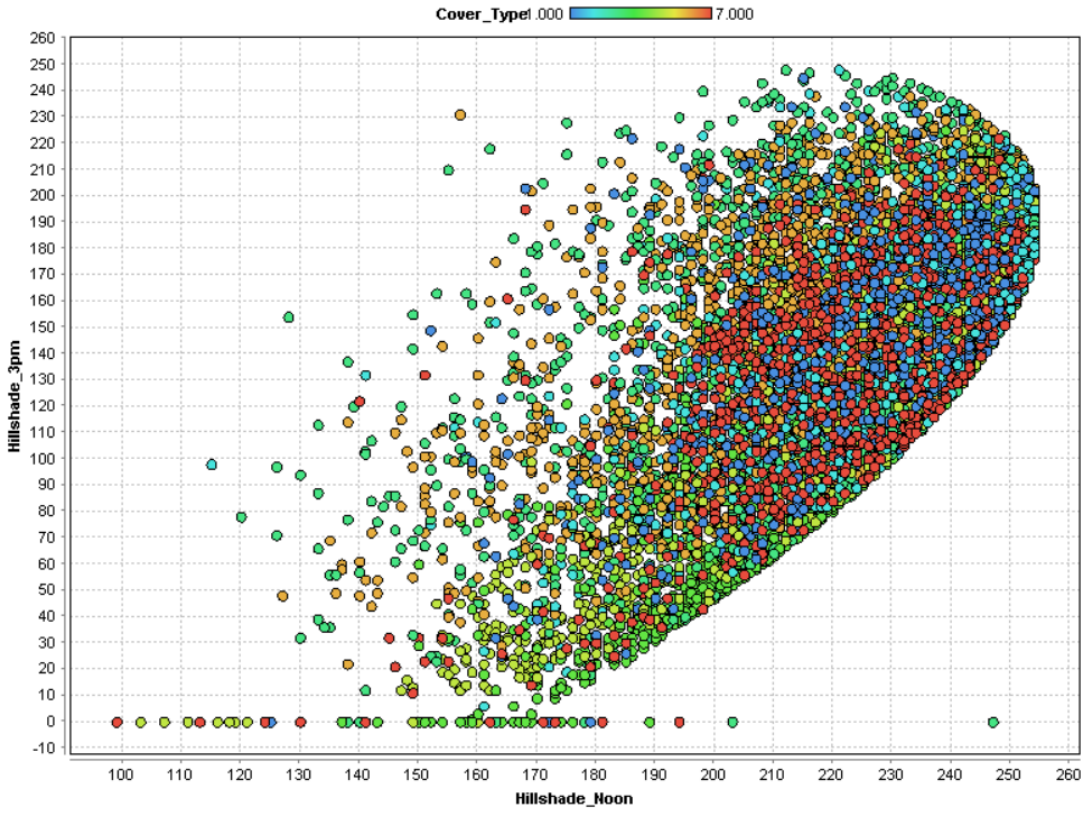
Rozložení classes
Určení class je poměrně jednoduché, vzhledem k tomu, co chceme dosáhnout, jsou classes jednotlivé Cover_Types, máme tedy 7 tříd do kterých můžeme data rozdělit. Tyto třídy jsou v původním datasetu poněkud nevyvážené, ale každá alespoň má rozumný počet instancí.

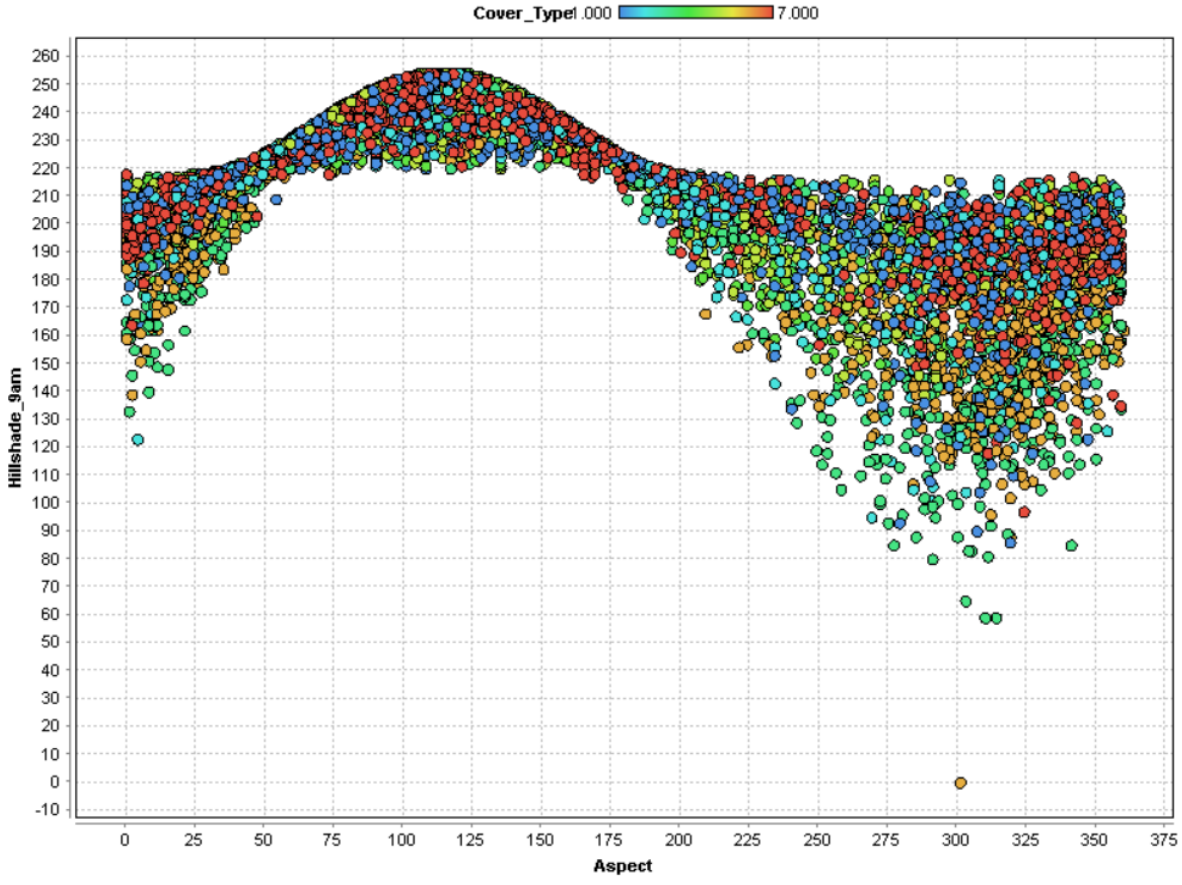
Import dat do matlab prostředí
Data jsem importovala z csv použitím funkce csvread('covtype.csv',1,0). 1 je druhým argumentem proto, že csv v sobě obsahuje hlavičky.

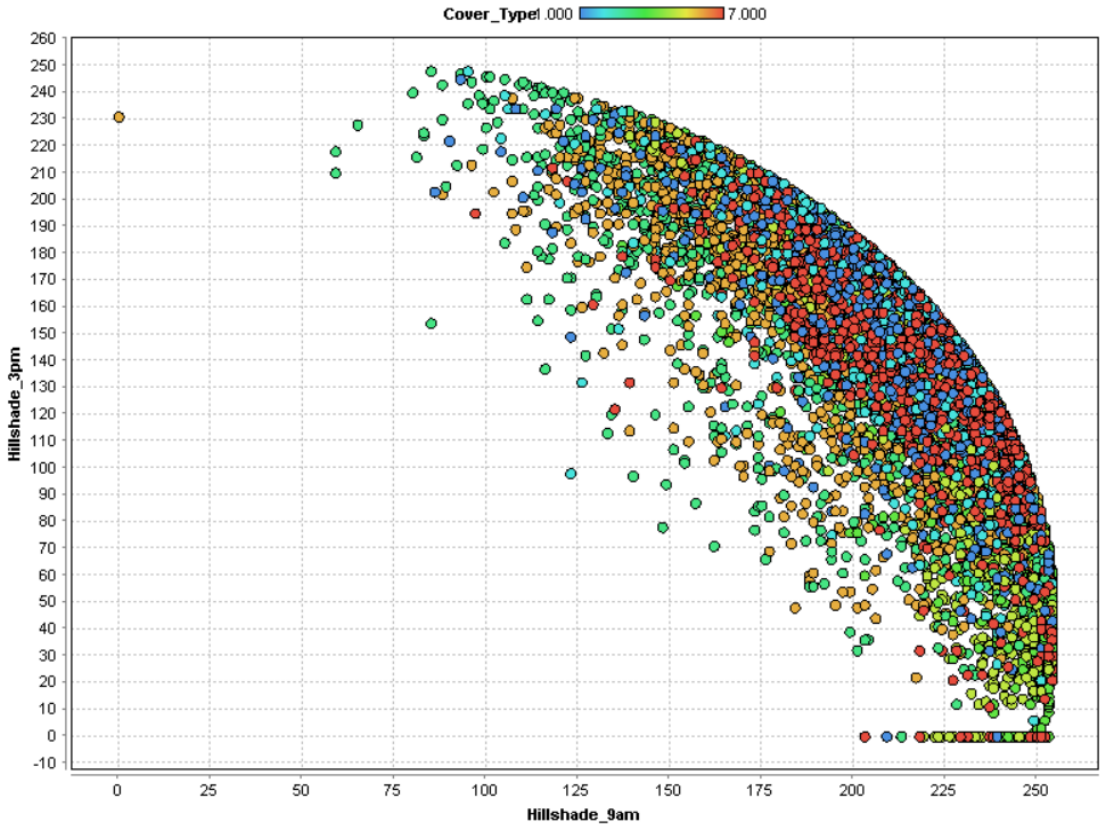
Korelace
Mezi těmito páry hodnot lze pozorovat poměrně silnou korelaci. Na následujících grafech můžeme vidět krásné elipsoidní vzory. Vertikální a horizontální vzdálenost vody pak dávají téměř lineární vzor.

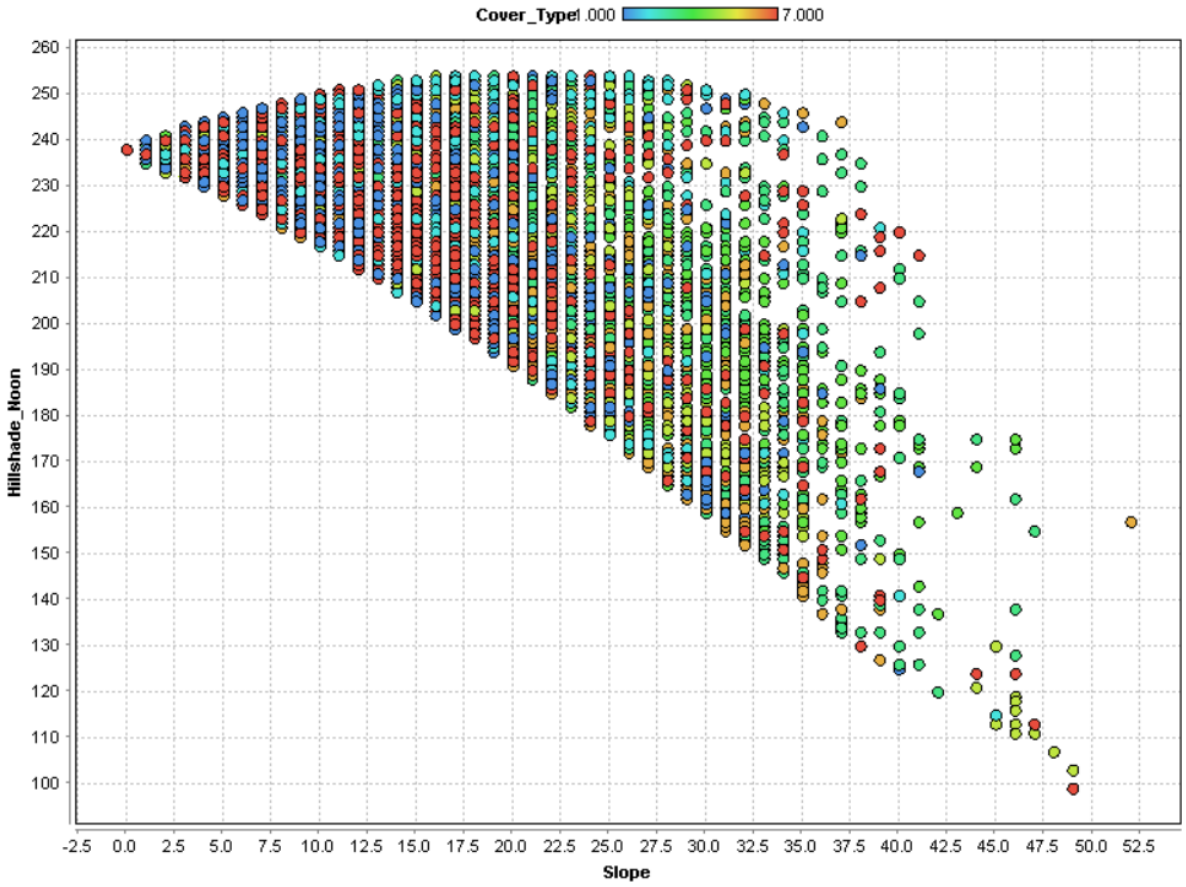
- Hillshade_9am a Hillshade_3pm = -0.78
- Aspect a Hillshade_3pm = 0.64
- Horizontal_Distance_To_Hydrology a Vertical_Distance_To_Hydrology = 0.60
- Hillshade_Noon a Hillshade_3pm = 0.59
- Aspect a Hillshade_9am = -0.58
- Slope a Hillshade_noon = -0.52




Cover_Type1.000
7.000
Hillshade_3pm
Hillshade_Noon


Cover_Type1.000
7.000
Hillshade_9am
Aspect





Outlier Removal

V této části se zabývám odstraněním takzvaných “bludných kamenů” neboli outliers, tedy hodnoty, které leží příliš daleko od průměru.
K nalezení bludných kamenů je možné použít několik různých metod v matlabu však pro ně existují speciální funkce již implementované.

- **isoutlier**
    - Toto je jednoduchá funkce, která projede vektor hodnot a označí všechny outliers. Pro matici pak projede každý řádek zvlášť. Výstupem je matice 1 a 0, které značí jestli hodnota na dané pozici je outlier nebo není. Moje data bylo tedy nejprve nutné transponovat: B = transpose(M).
    - Lze použít několik různých metod hodnocení, zde jsou výsledky mých dat na daných metodách:
        - **median** - vrací true pro ty hodnoty, které jsou blízko k mediánu podle MAD výpočtu (viz dokumentace)
            - sum(TF(:) == 1) => 7482787
            - sum(TF(:) == 0) => 24472873
        - **mean** - vrací true pro hodnoty, které mají standartní odchylku od průměru > 3, rychlejší, ale méně robustní
            - sum(TF(:) == 1) => 1151784
            - sum(TF(:) == 0) => 30803876
        - **quartiles** - vrací true pro prvky, které mají > 1.5 interquartile ranges nad horním quartilem, nebo pod dolním quartilem
            - sum(TF(:) == 1) => 7482787
            - sum(TF(:) == 0) => 24472873
        - **grubbs** - aplikuje Grubbsův test odlehlosti hodnot
            - sum(TF(:) == 1) => 7481139
            - sum(TF(:) == 0) => 24474521
        - **movmedian** - specifikuje okénko a aplikuje MAD vzdálenost
            - 3: sum(TF(:) == 1) => 2320071
                - sum(TF(:) == 0) => 29635589
            - 5: sum(TF(:) == 1) => 3121486
                - sum(TF(:) == 0) => 28834174
            - 6: sum(TF(:) == 1) => 3016831
                - sum(TF(:) == 0) => 28938829
            - 10: sum(TF(:) == 1) => 3037394
                - sum(TF(:) == 0) => 28918266
            - 100: sum(TF(:) == 1) => 7482787
                - sum(TF(:) == 0) => 24472873
- **hampel**
    - Aplikuje Hampelův filtr na vstupní vektor x, najde a odstraní outliers. Algoritmus spočítá odhad standartní odchylky každého prvku a pokud se prvek liší o více, než tři je odstraněná a nahrazená mediánem
    - Tady jsou výsledky nahrazených hodnot (0 - různé od původní matice, 1 - stejná hodnota jako původní hodnota)
    - sum(compareHampel(:) == 0) => 152490
    - sum(compareHampel(:) == 1) => 31803170

Vzhledem k výsledkům testů všemi metodami jsem se rozhodla nepoužít hampelovu metodu, přestože by se jevila jako ta nejméně pracná a data by zůstala relativně

nezměněná. Rozložení dat je normální, ale byla získána měřením a je třeba tedy vyfiltrovat hodnoty, které příliš nesedí. Pro nahrazení jsem se rozhodla jít zlatou střední cestou použitím funkce isoutlier s metodou movmethod s okénkem velikosti 3.

Pro nahrazení hodnot jsem pak vyzkoušela několik metod funkce filloutliers. Po zralé úvaze jsem se rozhodla použít nastavení fillmethod nearest, které nejlépe odpovídá rozložení mých dat a vycházelo ze všech metod nejlépe - neodstranila toho moc, ani málo.
sum(compareNew(:) == 1) => 29637775
sum(compareNew(:) == 0) => 2317885

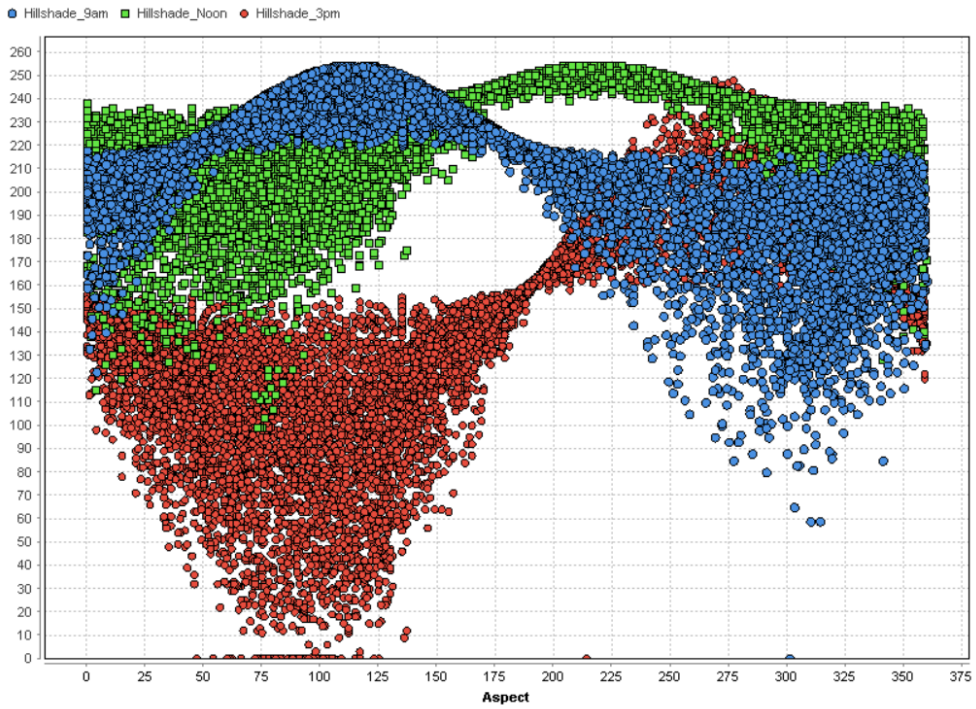
Konverze typu atributu
Vzhledem k tomu, že všechny atributy byly již v původních datech reprezentovány jako čísla, nebylo nutné nijak konvertovat atributy z textové reprezentace.

Diskretizace (binning)
Vzhledem k tomu, že hodnoty pocházejí z poměrně exaktního měření, jsem se rozhodla žádné údaje nediskretizovat. Diskretizací by vznikly nepřesnosti, kterým bych se ráda vyhnula a s údaji převedenými do intervalů by se hůře pracovalo.

Normalizace dat
Ohledně normalizace dat jsem přečetla asi tak tunu článků a různých rad jen ve snaze dozvědět se, jestli ji opravdu potřebuji. Většina názorů shodla na tom, že je lepší data normalizovat, pokud mezi nimi hledáme vztahy a pro lepší fungování některých algoritmů. Data jsem normalizovala do hodnot mezi 0 a 1 pomocí tohoto příkazu:

Xnorm = (M - min(M)) ./ (max(M) - min(M))

Feature selection
Vzhledem k tomu, že dataset má opravdu hodně features, bylo třeba je před samotnou prací probrat a zjistit, které je možné vyloučit, protože nejsou pro data podstatná.
Z analýzy korelace vyplívá, že všechny features Hill Shade spolu navzájem silně korelují. Také Aspect je s nimi spojený, především odpoledne. Také geografické faktory - elevation a slope spolu korelují a mají také vliv na druh porostu.



Protože data jsou velmi

široká, rozhodla jsem se redukovat počet features pomocí měření jejich importance. Data jsem si načetla do tabulky, následně použila regresní model fitrensemble pro přípravu dat a na základě toho pak určila důležitost jednotlivých features.

```
T = readtable("covtype.csv")
Mdl = fitrensemble(T,T(:,55))
imp = predictorImportance(ens)
```

Výsledky:

Elevation	0.6111	Soil_Type6	0.0016	Soil_Type24	0.0001
Aspect	0.0215	Soil_Type7	0	Soil_Type25	0
Slope	0.0062	Soil_Type8	0	Soil_Type26	0.0005
Horizontal_Distance_To_Hydrology	0.0244	Soil_Type9	0	Soil_Type27	0.0002
Vertical_Distance_To_Hydrology	0.0287	Soil_Type10	0.0041	Soil_Type28	0.0012
Horizontal_Distance_To_Roadways	0.0949	Soil_Type11	0.0022	Soil_Type29	0.0052
Hillshade_9am	0.0141	Soil_Type12	0.0007	Soil_Type30	0.0081
Hillshade_Noon	0.0174	Soil_Type13	0.0027	Soil_Type31	0.0672
Hillshade_3pm	0.0118	Soil_Type14	0.0011	Soil_Type32	0.0036
Horizontal_Distance_To_Fire_Points	0.1112	Soil_Type15	0	Soil_Type33	0.0007
Wilderness_Area1	0.0465	Soil_Type16	0.0003	Soil_Type34	0.0055
Wilderness_Area2	0.0307	Soil_Type17	0.0046	Soil_Type35	0.0001
Wilderness_Area3	0.0557	Soil_Type18	0.0003	Soil_Type36	0.0065
Wilderness_Area4	0.0018	Soil_Type19	0.0002	Soil_Type37	0.0171
Soil_Type1	0.0004	Soil_Type20	0.0010	Soil_Type38	0.0542
Soil_Type2	0.0051	Soil_Type21	0.0004	Soil_Type39	0.0125
Soil_Type3	0.0003	Soil_Type22	0.0082	Soil_Type40	0.0018
Soil_Type4	0.0046	Soil_Type23	0.0032		
Soil_Type5	0.0004				

Z výsledků je jasné, že moje vizuální interpretace byla poměrně přesná. Elevation se zdá být tou nejdůležitější feature pro všechny typy porostu. Aspect indikuje, kterým směrem na daném místě jsou rostliny obrácené, což jistě ovlivňuje kolik slunce strom dostane. Není proto překvapením, že tato feature je také označena jako důležitá.
Důležitost přítomnosti vody je neoddiskutovatelná. Naopak mě překvapilo, jak málo podstatný je druh půdy.
Na základě těchto výsledků jsem se rozhodla zredukovat počet features z 55 na 12. Vyloučila jsem všechny Soil_Types, Wilderness_Area4 a Slope, které mají na výslednou class jen malý vliv. Tato redukce by pak měla velmi pomoci rychlosti sestavení modelu.

Vlastní práce

Protože předpovídání, jaký typ porostů bude v jakém místě není zrovna příliš složitou úlohou, rozhodla jsem se vyzkoušet několik různých klasifikačních metod. Výstupy by pak mohly sloužit jako například základ pro generování realistických lesů do video her, nebo modelů parku. Možné využití by také bylo při umělé obnově krajiny parku, kdy by se dalo předpovědět, jestli vysazené stromy v daném místě vydrží a jaké druhy sázet kam.

Náhodný výběr dat do datasetů

Pro učení a testování modelu jsem se rozhodla pro oba datasety (testovací i tréninkový) vybrat náhodné hodnoty s opakováním. Poměr testovacích a tréninkových dat je 25% ku 75% Tento výběr jsem provedla pomocí posloupnosti těchto příkazů:

```
k = randperm(581012);
MxTest = NewM(k(1:2500),:);
k = randperm(581012);
MxTrain = NewM(k(1:7500),:);
```

K těmto datasetům jsem následně určila labely s využitím klasifikace do 7 tříd Cover_Type:

```
MyTest = MxTest(:,13)
MyTrain = MxTrain(:,13)
```

Následně jsem labely smazala z datasetů x.

Cross-validace

Cross-validaci jsem se rozhodla pouze vyzkoušet na některých modelech, abych viděla jak to dopadne. Jak to dopadlo je vidět na následujících datech, která byla naměřena s použitím různých algoritmů pro cross-validaci. Pro predikci pomocí cross validace jsem použila metodu kfoldPredic, která vrací labely class předpovězené v modelu - pomocí cross valid klasifikace.

Mdl = fitcensemble(Tnew,MyTrain(:,1),'Method','Bag')
[elabel,escore] = kfoldPredict(CVMdl);
max(escore) 0.9900 1.0000 0.9900 0.7100 0.8900 0.9200 0.9600

Mdl = fitcensemble(Tnew,MyTrain(:,1),'Method','AdaBoostM2')
[elabel,escore] = kfoldPredict(CVMdl);
max(escore) 4.7094 4.8863 3.5170 2.6307 3.2430 3.2491 4.2635
min(escore) 0 0.4282 0 0 0 0 0

Mdl = fitcensemble(Tnew,MyTrain(:,1),'Method','LPBoost')
[elabel,escore] = kfoldPredict(CVMdl);
max(escore) -0.1395 -0.1046 -0.2065 -0.3630 -0.3378 -0.2410 -0.1674
min(escore) -0.9855 -0.8237 -1.0000 -1.0000 -0.9926 -1.0000 -1.0000

Klasifikační stromy

Jako další metodu predikce jsem použila více klasifikačních stromů. V následujicí části jsou i s výsledky accuracy pro předpověď.

- **fitctree**
  - fitctr = fitctree(MxTrain,MyTrain)
  - accuracy předpovědi = 0.7450
- **fitrtree**
  - fittr = fitrtree(MxTrain,MyTrain)
  - accuracy předpovědi = 0.5310

Klasifikační modely

Pro předpovídání typu porostů jsem si vybrala několik různých klasifikačních modelů, které jsou v zmíněné jako nejpoužívanější, následně můžeme vidět výsledky jejich předpovídání pro testovací data.

- **Discriminant Analysis Classification**
  - MdlLinear = fitcdiscr(MxTrain,MyTrain)
  - accuracy předpovědi = 0.6940
- **SVM**
  - jelikož mám více, než dvě třídy dat (celkem 7), musela jsem použít funkci fitcecoc
  - Mdl = fitcecoc(MxTrain,MyTrain)
  - accuracy předpovědi = 0.2660
- **KNN**
  - **Md=fitcknn(MxTrain,MyTrain,'Distance','cityblock')**
    - accuracy předpovědi = 0.8270
  - **Md=fitcknn(MxTrain,MyTrain,'Distance','chebychev')**
    - accuracy předpovědi = 0.7920
  - **Md=fitcknn(MxTrain,MyTrain,'Distance','correlation')**
    - accuracy předpovědi = 0.6810
  - **Md=fitcknn(MxTrain,MyTrain,'Distance','cosine')**
    - accuracy předpovědi = 0.6880
  - **Md=fitcknn(MxTrain,MyTrain,'Distance','euclidean')**
    - **NumNeighbors 1**
      - accuracy předpovědi = 0.8180
    - **NumNeighbors 3**
      - accuracy předpovědi = 0.7910
    - **NumNeighbors 5**
      - accuracy předpovědi = 0.7870
    - **NumNeighbors 10**
      - accuracy předpovědi = 0.7710
    - **NumNeighbors 50**
      - accuracy předpovědi = 0.7150

        - **Md=fitcknn(MxTrain,MyTrain,'Distance','hamming')**
            - accuracy předpovědi = 0.4520
        - **Md=fitcknn(MxTrain,MyTrain,'Distance','jaccard')**
            - accuracy předpovědi = 0.4510
        - **Md=fitcknn(MxTrain,MyTrain,'Distance','mahalanobis')**
            - accuracy předpovědi = 0.7980
        - **Md=fitcknn(MxTrain,MyTrain,'Distance','minkowski')**
            - accuracy předpovědi = 0.8180
        - **Md=fitcknn(MxTrain,MyTrain,'Distance','seuclidean')**
            - accuracy předpovědi = 0.7810
        - **Md=fitcknn(MxTrain,MyTrain,'Distance','spearman')**
            - accuracy předpovědi = 0.4920
- **Native Bayes**
    - u nativního bayese jsem byla nucena použít jiný kernel, než je defaultní, protože třída 3 mých dat má nulový rozptyl
    - **Md=fitcnb(MxTrain,MyTrain, 'DistributionNames', 'kernel','Kernel', 'box')**
        - accuracy předpovědi = 0.6890
    - **Md=fitcnb(MxTrain,MyTrain, 'DistributionNames', 'kernel','Kernel', 'epanechnikov')**
        - accuracy předpovědi = 0.6860
    - **Md=fitcnb(MxTrain,MyTrain, 'DistributionNames', 'kernel','Kernel', 'normal')**
        - accuracy předpovědi = 0.6900
    - **Md=fitcnb(MxTrain,MyTrain, 'DistributionNames', 'kernel','Kernel', 'triangle')**
        - accuracy předpovědi = 0.6880

Interpretace výsledků

Žádným překvapením není nízký výsledek SVM. SVM algortimy mají obvykle problém na nevybalancovaných datech. V mém případě v datech převládá jedna třída (2), což přispívá k nepřesnosti tohoto algoritmu. Tento problém by bylo možné odstranit lepším než náhodným výběrem testovacích a tréninkových datasetů, nicméně bych si stejně moc nepomohla. SVM není dobrý algoritmus pro zacházení s více třídami.

Očekávaný byl také dobrý výkon klasifikačního stromu. Rozhodovací stromy jsou všeobecně vhodné k předpovědím ve vysoce nelineárních modelech s komplexními vztahy, kde se z jejich potenciálu dá vytěžit nejvíce.

Z výsledků předchozích zjištění je zřejmé, že nejlepší algoritmus pro klasifikaci a předpovídání nad těmito daty je KNN. Řekla bych, že KNN velmi profitovalo z normalizace dat, kterou jsem provedla před jeho spuštěním. Poněkud mě překvapilo, že nejlépe dopadla vzdálenost cityblock. V závěsu za ní se po té drží také vzdálenost minkowski a euklidovská vzdálenost s počtem sousedů 1, které se liší - pokud jsem to správně pochopila - jen v tom, že berou v úvahu jiný počet dimenzí.