Rapport

# Inledning

Detta är en rapport på en end-to-end machine learning-projekt för att prediktera befolkningsmängd på en viss punkt i Kalifornien.

## 1.1 Bakgrund

Till detta projekt har jag använt datasetet california\_housing som är ett välkänt dataset för att prediktera husvärden. Varje instans i datasetet är ett ”block” på kartan.

*” This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S.Census Bureau publishes sample data (a block group typically has a populationof 600 to 3,000 people”*

(https://inria.github.io/scikit-learn-mooc/python\_scripts/datasets\_california\_housing.html)

## Syfte och frågeställning

Då datasetet verkar vara mest använt för att prediktera husvärden har detta projekt som syfte att se om det även går att prediktera något annat än just husvärden på samma data.

Min frågeställning blir då: *Går det att utifrån datasetet prediktera hur tätbefolkat ett block är?*

# Databeskrivning / EDA (Exploratory Data Analysis)

Datasetet innehåller 20640 rader och 10 kolumner. Alla datatyper är float64 förutom kolumnen ocean\_proximity som är ett text-objekt. Den kolumn som jag ska prediktera är population och den har värden som sträcker sig mellan 3 – 35682. Datasetet innehåller 207 saknade värden i kolumn total\_bedroom.

Jag har valt att använda mig av ett valideringset för att välja min slutgiltiga modell, därför har jag delat upp housing-datasetet i tränings-, validering- och testset. Träningsdatan består av 13209 instanser, valideringsdatan 3303 och testdatan består av 4128 instanser. Jag har även ett större träningset som är tränings- och valideringsdatan kombinerat, den används endast till den bästa modellen.

En bild som visar skärmbild, karta, text

Automatiskt genererad beskrivning

Efter att ha plottat ut populationen med hjälp av longitud och latitud kan jag se att det finns kluster där populationen är större. Detta stärker min teori om att det går att prediktera befolkningsmängd med hjälp av detta dataset.

# 3. Metod och Modeller

## 3.1 Metoder

Jag har processat datan innan jag låter modellerna träna. Då kolumnen ocean\_proximity innehåller objekt och inte numeriska värden använder jag Pandas *get\_dummies* för att One-Hot-Encoda värdena till nominala värden. Kolumnen total\_bedrooms saknar värden och jag valde att angripa detta genom att fylla i kolumens median genom en *SimpelImputer*. Jag skalade därefter alla X-värden med hjälp av *StandardScaler.* Med hjälp av *GridSearchCV* har jag, i tre folds, letat efter de hyperparametrar i varje modell som lämpar sig bäst för problemet. Då jag har tagit mig an ett regressionsproblem har jag också valt regressionsmodeller.

## 3.2 Lasso

Lasso är en linjär regressionsmodell som har hyperparametern alpha. Alpha är en regulariserare som gör att variansen minskar och gör modellen mindre flexibel.

## 3.3 RandomForestRegressor

RandomForestRegressor är en form av ensamble learning bestående av flera desicion trees. Desicion trees är uppbyggt som ett lövverk på det träd. Den strävar efter att ha ”rena” löv och fortsätter att förgrena sina noder tills den når ett löv.

## 3.4 ExtraTreesRegressor

ExtraTreesRegressor är en variant av RandomForest men har också en randomiserad threshold för varje nod.

4. Projekt Resultat och Analys

Modellernas bästa hyperparametrar är:

|  |  |
| --- | --- |
| **Modell** | **Hyperparametrar** |
| Lasso | Alpha=0.1 |
| RandomForestRegressor | Max\_depth= 25, n\_estimators=100 |
| ExtraTreesRegressor | Max\_depth=30, n\_estimators=150 |

Lassos bästa alpha är 0.1 vilket antyder att det ligger nära en vanlig linjär regression.

Jag har jämfört mina modeller utifrån Root Mean Square Error:

|  |  |
| --- | --- |
| **Modellutvärdering** | |
| **Modell** | **RMSE** |
| Lasso | 422.5 |
| RandomForestRegressor | 394.3 |
| ExtraTreesRegressor | 390.6 |

Eftersom man vill ha ett så låg RMSE som möjligt valde jag att gå vidare med ExtraTreesRegressor som fick det bästa resultatet. Efter att ha tränat om denna modell på både tränings- och valideringssetet fick modellen prediktera på testsetet och fick då resultatet nedan.

|  |  |  |
| --- | --- | --- |
| **Modell** | **RMSE** | **Procent fel** |
| ExtraTreesRegressor | 409.5 | 0.29 |

En RMSE på ca 400 kan anses vara lågt men då värdena har ett relativt litet spann (3 - ~35000) i population gör att RMSE kommer vara lågt även vid de lite sämre modellerna. Därför var det också viktigt för mig att få ut antal fel modellen predikterar i procent. Jag upplever att det då blir lättare att sätta värdet i ett sammanhang.

Jag ställdes inför olika val under processen. Exempelvis rörde det kolumnen ocean\_proximity, som är ett text-objekt. Då maskininlärningsalgoritmerna kräver numerisk data har jag valt att One-hot-encoda ocean\_proximity. En lösning hade annars kunnat vara att ta bort hela kolumnen men jag tror att blockets område är en viktig information och jag har valt att behålla den. Jag hade också kunnat välja att göra en ordinal encoding på denna kolumn men eftersom jag inte vet vilket område som anses ha högre värde än den andra vet jag inte hur jag skulle kunna rangordna dom. Att ha en nominal encoding på dessa värden gör att jag låter modellerna hitta mönster på vilket sätt blockets område påverkar befolkningsmängden.

En bild som visar text, skärmbild, meny, Teckensnitt

Automatiskt genererad beskrivningDet var i börjar en osäkerhet för mig om det gick att prediktera befolkningsmängd och därför valde jag att ha kvar alla kolumner i datasetet, så att det fanns så mycket data som möjligt för modellerna att lära sig från. Utifrån min korrelationsmatris kan jag se att starkt, positivt, linjärt samband mellan population och de kolumner som rör hushåll, antal sovrum och antal rum.

# 5. Slutsats och förslag på potentiell vidareutveckling

Utifrån modellernas resultat menar jag att det går att prediktera befolkningsmängd på california\_housing. Jag är nöjd med att ha fått en felmarginal på ca 30%.

Som en potentiell vidareutveckling hade jag kunnat klassificera ett block som tätbefolkat eller icke-tätbefolkat genom att sätta thresholds för de olika klasserna: tätbefolkat samt glesbefolkat.