Universiteit van Amsterdam

Leren en beslissen

Criminaliteitsdata als voorspeller van huizenprijzen

Januari 2024

Studenten:
Abe Dijkstra
14622556

Florence Lont
14122723

Puck Voorham
13419900

Vera Westerman
13197444

Begeleider:
David Meertens

Opdrachtgever:
KR&A

Docent:
Maas Hermes

Cursus:
Leren & Beslissen

Abstract

Dit onderzoek betreft het voorspellen van de vraagprijs voor Nederlandse huur- en koophuizen. Dit is gedaan op basis van traditionele data in combinatie met de alternatieve data van inbraakscijfers. De opdracht is gegeven door het vastgoedadviesbureau KR&A dat beleggers advies geeft op basis van data. De traditionele data bestaat uit eigenschappen van de huizen volgens huisverkoop websites met attributen zoals locatie, huisoppervlakte en het bouwjaar. De gegevens zijn afkomstig uit een door KR&A geleverde dataset. Dit is gecombineerd met openbare gegevens over inbraken in de gemeente en het postcodegebied, afkomstig van de politie. Er wordt gekeken in hoeverre criminaliteitscijfers bij kunnen dragen aan het verbeteren van kunstmatig intelligente modellen die huizenprijzen kunnen voorspellen. De gebruikte modellen zijn *Ridge Regression*, *Lasso Regression*, *Decision Tree Regression*, *Multilayer Perceptron Regression*, *Deep Artificial Neural Network Regression*, *Nearest Neighbours Regression*, *Support Vector Regression*, *XGBoost Regression*, *AdaBoost Regression*, *Gradient Boost Regression Random Forest Regression*, *Bagging Regression*, *Voting Regression* en *Stacking Regression*. Het best werkende eindmodel is *Random Forest Regression* met een *mean absolute error (MAE)* van 13.8% voor Nederland en 11.4% voor Rotterdam. Er is vergeleken of de toevoeging van de inbraakcijfers de modellen beter maakt. Dit was bij de landelijke modellen niet het geval, maar bij het uitvoeren op steden op zichzelf was er verbetering zichtbaar; van een MAE van 11.4% naar een MAE van 11.1% voor Rotterdam.

Inhoudsopgave

1 Probleemstelling

De Nederlandse woningmarkt kenmerkt zich vandaag de dag door sterk beperkte beschikbaarheid en hoge huizenprijzen. Het inzichtelijk maken van de diverse factoren die deze markt beïnvloeden kan van groot belang als men een beter zicht wilt op deze huizenmarkt (Cox, Brounen & Neuteboom, 2015). Dit onderzoek richt zich op het voorspellen van huizenprijzen met behulp van verschillende *machine learning* technieken. Hiervoor is opdracht gegeven door het bedrijf KR&A. Zij bieden investeerders en financiers oplossingen voor verbeterde rendementen en beter risicobeheer voor vastgoedfondsen en -bedrijven in heel Europa. Dit wordt gedaan door middel van advies over data, strategie en duurzaamheid (KR&A, 2023). KR&A specialiseert zich voornamelijk in het gebruik van alternatieve data. Dit is niet-traditionele data, wat inhoudt dat het niet de data is die voor de hand ligt bij het voorspellen van huizenprijzen. In dit onderzoek zijn, aan de hand van een door KR&A verstrekte database, voorspellingen gedaan over de huizenprijzen. Hierbij zijn traditionele variabelen uit deze dataset, zoals locatie, oppervlakte en bouwjaar, in beschouwing genomen. Dit is gecombineerd met alternatieve data van de Politie (Politie, 2024) met gegevens over woninginbraak per gemeente. Met als doel om betere voorspellingen te maken van de huizenprijzen met behulp van de inbraakgegevens en daarbij ook meer inzichten te bieden in de effecten van criminaliteit op de huizenmarkt.

De onvoorspelbaarheid en complexiteit van vastgoedmarkten vormen aanzienlijke uitdagingen voor woningzoekenden en bedrijven die betrokken zijn bij vastgoedtransacties. Traditionele methoden voor het schatten van woningwaarden schieten vaak tekort bij het vastleggen van de ingewikkelde patronen en dynamische factoren die van invloed zijn op woningprijzen. Kunstmatige intelligentie biedt kansen om betere voorspellingen te doen over de waarde van huizen en de huurprijzen en is sterk aan het opkomen op dit gebied (Funda, 2023). Hierbij wordt al vaak gewerkt met de traditionele data, maar juist de combinatie met alternatieve data wordt op dit moment ondergewaardeerd. Op deze manier kunnen nog niet eerder gevonden verbanden worden geïdentificeerd. Deze verbanden kunnen de voorspellingen van prijzen op een unieke manier verbeteren en bieden klanten van KR&A een nieuw perspectief op de huizenmarkt. Dit onderzoek richt zich op de alternatieve data van criminaliteitscijfers, met een specificatie op de aantallen woninginbraken. Er wordt gekeken in hoeverre criminaliteitscijfers bij kunnen dragen aan het verbeteren van kunstmatig intelligente modellen die huizenprijzen kunnen voorspellen.

Criminaliteitscijfers zijn een duidelijke numerieke afspiegeling van de leefomgeving en veiligheid in een buurt. Het niveau van criminaliteit in een bepaald gebied beïnvloedt de perceptie van veiligheid, wat een cruciale overweging voor potentiële kopers en investeerders is. Zo is er in een onderzoek van O'Sullivan (O'Sullivan, 2000) gevonden dat criminaliteit de huizenprijzen in stadscentra negatief beïnvloeden. Maar ook andersom is aangetoond dat economische ontwikkeling in een buurt wordt beperkt door criminaliteit (Speller, Pol & Mingardo, 2006) . Hierdoor is het van belang meer onderzoek te doen naar de effecten van criminaliteit op de huizenprijzen in Nederland. In dit onderzoek is specifiek ingegaan op de woninginbraken per postcodegebied, aangezien dit over het algemeen elke bewoner kan overkomen en daarom direct in verband staat met de veiligheid van het wonen in een bepaalde buurt. Door deze data te integreren in de voorspellingsmodellen van de huizenprijzen wordt de impact van veiligheidsfactoren op de vastgoedmarkt onderzocht.

2 Data

2.1 Dataset KR&A

2.1.1 Beschrijving

Vanuit KR&A zijn drie datasets aangeleverd. Deze bevatten advertenties van woningen die in de maanden oktober, november en december te koop of te huur zijn aangeboden op drie belangrijke woningwebsites. Deze websites zijn Funda, Pararius en Jaap. Samen bestaan de drie datasets uit 329.076 datapunten; het totale aantal advertenties. De datapunten zijn redelijk gelijkmatig verdeeld over heel Nederland. Er zijn geen grote gebieden waar wel huizen staan maar dekking van deze dataset mist.

De dataset bevat 35 kolommen die verschillende eigenschappen van de advertenties bevatten. Dit gaat om eigenschappen over de website waar de advertentie vandaan komt, de prijs, onderdelen die inbegrepen zijn bij de prijs, het bouw- en renovatiejaar, de oppervlakte, overige specificaties en de adresgegevens en locatie. De data beschikt dan ook over zowel numerieke variabelen zoals de huur- of aankoopprijs als categorische variabelen zoals de stadsnaam. In tabel 5 staan deze eigenschappen, voor de duidelijkheid ingedeeld in categorieën.

Tabel 1:

Categorie	**Features in dataframe (numerieke features vetgedrukt)**
Websitegegevens	source, sourceURL, sourceId, referenceDate, broker
Gegevens over de prijs	**askingprice**, including purchase cost, service charges, services included
Gegevens over het huis	property type, **construction year**, **renovation year**, **living area**, **land area**, **number of rooms**, energy label, description, particularities, construction type, new construction
Locatiegegevens	address street, address house number, address house number add, address city, address zip code, address construction number, address inaccurate address, address country, **geodata latitude**, **geodata longitude**, geodata precise

2.1.2 Opschoning

De aangeleverde datasets zijn overzichtelijk, maar bevatten veel missende waarden, afhankelijk van de website waarop de woning werd geadverteerd. In tabel 2 staan de aantallen verwijderde datapunten per categorie. Als eerste zijn alle datapunten zonder de *target variable* (de vraagprijs) verwijderd. Vervolgens zijn alle huizen die er dubbel instonden geselecteerd, dit is bepaald op basis van de `sourceID`. Alleen de meest recente van elk van de duplicaten is behouden. Mochten huizen bijvoorbeeld opnieuw te koop of te huur zijn aangeboden, omdat ze de eerste maand nog niet verkocht waren, dan komt de laatste prijs vermoedelijk het beste overeen met de echte prijs, die wij willen voorspellen. Daarna zijn de huizen zonder oppervlakte verwijderd, aangezien alle modellen de oppervlakte of de prijs per vierkante meter gebruiken. Na het visualiseren van de data, bleken er een aantal uitschieters te bestaan in de advertenties in prijs-oppervlakte verhouding. Dit bleken advertenties van bijvoorbeeld objecten als parkeergarages of percelen. Er is voor gekozen om alle advertenties die niet werden bestempeld als huis, te verwijderen uit de dataset. Dit is gedaan door alle datapunten waarbij de woorden 'parkeergelegenheid', 'object', 'overige', 'bouwgrond' of 'recreatie' in de `sourceId` voorkwamen, te verwijderen.

Ook de nieuwbouwprojecten zijn op deze wijze verwijderd. Bij deze punten was de vraagprijs namelijk als een geschat bereik neergezet en ging het niet om een enkele prijs. Alle datapunten met 'bouwperceel' in de beschrijving zijn ook verwijderd. Als laatste zijn vier incorrecte datapunten verwijderd. Deze vielen bij de datavisualisatie op als uitschieters. Het gaat hierbij om drie huizen die een bouwjaar rond het jaar 1000 hebben, terwijl het oudste Nederlandse woonhuis uit 1130 komt (Kreek & Slechte, 2020). En er was een huis waarbij stond dat het `propertyType` *'other offer'* was. Ze zijn allen handmatig gecontroleerd op bruikbaarheid en er bleek dat ook andere gegevens niet correct waren.

Tabel 2: Gefilterde datapunten

Verwijderde categorie	**Aantal verwijderde datapunten**
Datapunten zonder vraagprijs	12.816
Dubbele huizen	154.863
Datapunten zonder oppervlakte (binnenshuis)	4.714
Objecten of nieuwbouwprojecten	2.522
Incorrecte datapunten (handmatig gecontroleerd)	4

Het opdelen van de data in huur- en koopwoningen wordt gedaan door in de URL op de termen 'te-koop', 'koop', 'te-huur' of 'huur' te zoeken. Uit een test bleek dat dit accurater is dan om de boolean `IsSale` te gebruiken, omdat deze niet altijd correct van de websites is overgekomen. Deze opsplitsing is essentieel, aangezien de vraagprijs van een koophuis erg verschilt van die van een huurhuis. Na de opschoning van de data is het onderzoek met de volgende aantallen huur- en koophuizen uitgevoerd: (zie tabel 3).

Vervolgens zijn enkele waarden omgezet naar numerieke data, met als doel om de kolommen bruikbaar te maken voor de modellen. Veel modellen kunnen namelijk alleen cijfers gebruiken. Zo zijn de letters en woorden in de kolom *energy label* aangepast naar cijfers, waarbij 1 = A+++++ en 12 = G. Op deze manier konden de energielabels van verschillende websites worden vergeleken, aangezien sommige energielabels erin stonden met een het woord PLUS en anderen met het teken '+'. Verder is de kolom *property type* veranderd in *binary values*, alleen de variabele *house* (=0) en *apartment* (=1) stonden in deze kolom. Verder zijn alle *NaN-values* (plekken waar data mist) in de kolom *renovation year* vervangen door een 0 en daarna zijn alle plekken waar wel een jaar stond, vervangen door een 1. De kolom betekent nu dus of er bekend is of er een renovatie heeft plaats gevonden of niet.

2.2 Alternatieve data

Er is naast de data van KR&A ook gebruik gemaakt van de data van woninginbraken binnen Nederland, afkomstig van de politie database (Politie, 2024). Deze data is openbaar beschikbaar en kan een nieuwe inkijk geven aan het probleem. De data bestaat uit woninginbraken en pogingen tot woninginbraak per postcodegebied van de afgelopen drie maanden (vanaf 23 oktober tot 23 januari 2024). De politie vermeldt individuele meldingen van woninginbraak per datum en postcodegebied. Met behulp van de *requests library* (Reitz, K., 2023), kunnen de gegevens van een website, door middel van de URL, in Python worden ingeladen. De website van de politie biedt deze data echter enkel per stad aan. Met behulp van een loop door alle steden die in de dataset van KR&A voorkomen, is een *pandas DataFrame* gemaakt, waarin er een kolom is met

	Totaal	**Met postcode**
Koophuizen	137.630	106.611
Huurhuizen	16.707	8.000

Tabel 3: Het aantal datapunten na de opschoning van de data

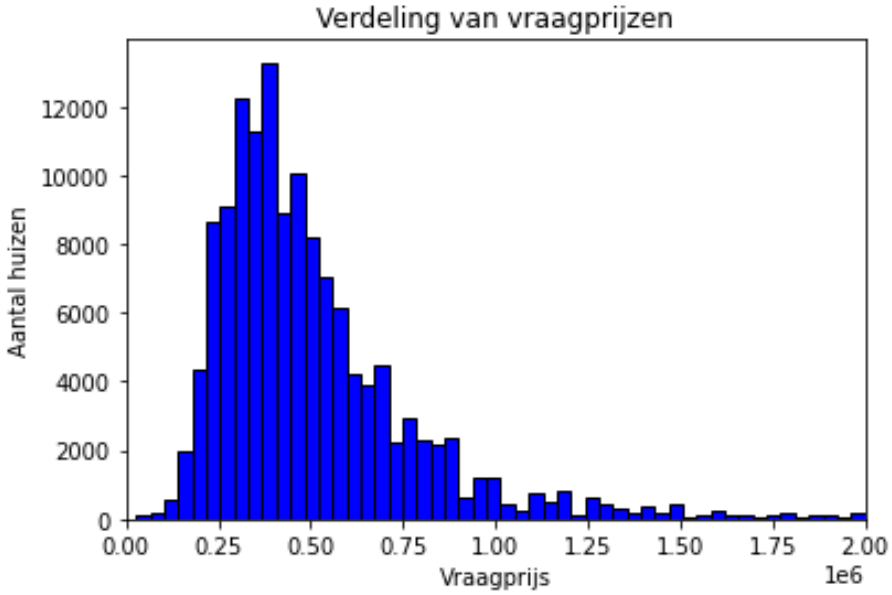
postcodes en een kolom met 'inbraak' en 'poging tot inbraak'.

Hierna is het aantal inbraken en pogingen tot inbraak bij elkaar opgeteld per postcode. Deze dataset is vervolgens samengevoegd met de dataset van KR&A, waarbij voor elk huis het aantal inbraken en pogingen tot inbraak in het betreffende postcodegebied (alleen het nummer, zonder letters) is toegevoegd aan de nieuwe kolom `number_of_crimes`. Echter is deze data dus niet beschikbaar voor datapunten zonder postcode. Bij de vergelijking van de modellen zonder en met de criminaliteit zijn deze datapunten zonder postcode niet gebruikt. Voor de koop zijn dit er 31.019 en voor de huur 8.707.

2.3 Datavisualisatie

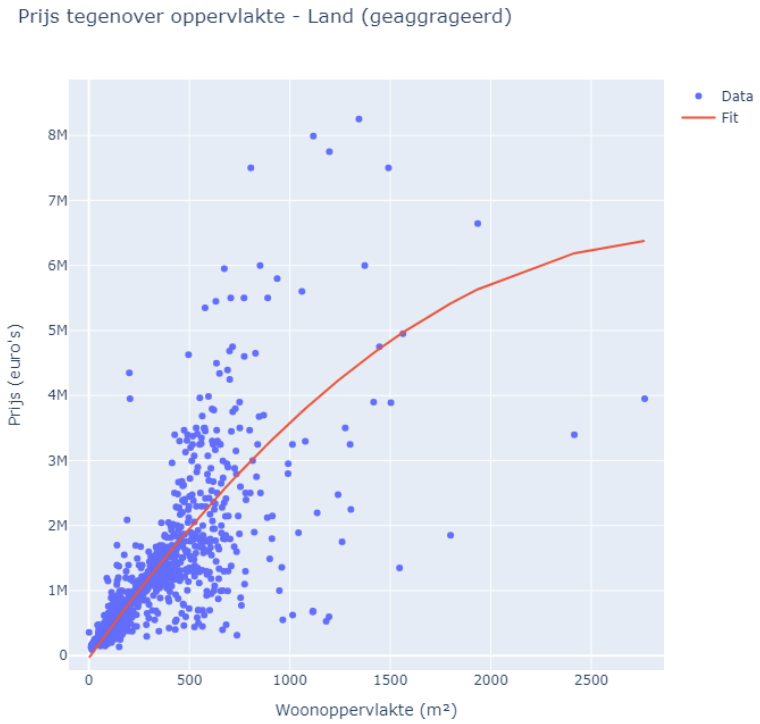
2.3.1 Datasets KR&A

Om een duidelijk beeld van de dataset van KR&A te krijgen, zijn er verschillende visualisaties gemaakt. Deze grafieken en diagrammen bieden een visuele weergave van de belangrijkste patronen, trends en verbanden in de data. Deze visualisaties zijn gemaakt met behulp van de Plotly, Matplotlib en Seaborn. Ten eerste is er gekeken naar de verdeling van de prijzen van de koophuizen in de dataset. In figuur1 is zichtbaar dat ruim de meeste huizen tussen de 250.000 en 500.000 euro kosten. Verder zijn er een redelijk aantal huizen met een prijs boven de 1 miljoen.



Figuur 1: Kaart met de prijs van koopwoningen

Daarnaast is ook de vraagprijs tegenover de woonoppervlakte uitgezet. In figuur 2 is dit verband voor de stad Rotterdam zichtbaar. Hierbij is een polynomiale regressie uitgevoerd om de relatie tussen de prijs en het oppervlakte duidelijk te krijgen. Uit deze grafiek valt op te maken dat er bij huizen onder de drie miljoen een vrijwel lineair verband bestaat tussen de prijs en de oppervlakte. Dit is ook het gebied waar de meeste huizen in liggen. Gezien dit vrijwel lineaire verband is er voor gekozen om in rest van het onderzoek door te werken met de prijs per vierkante meter. Op deze manier kan de woonoppervlakte gelijk in alle modellen mee worden genomen.



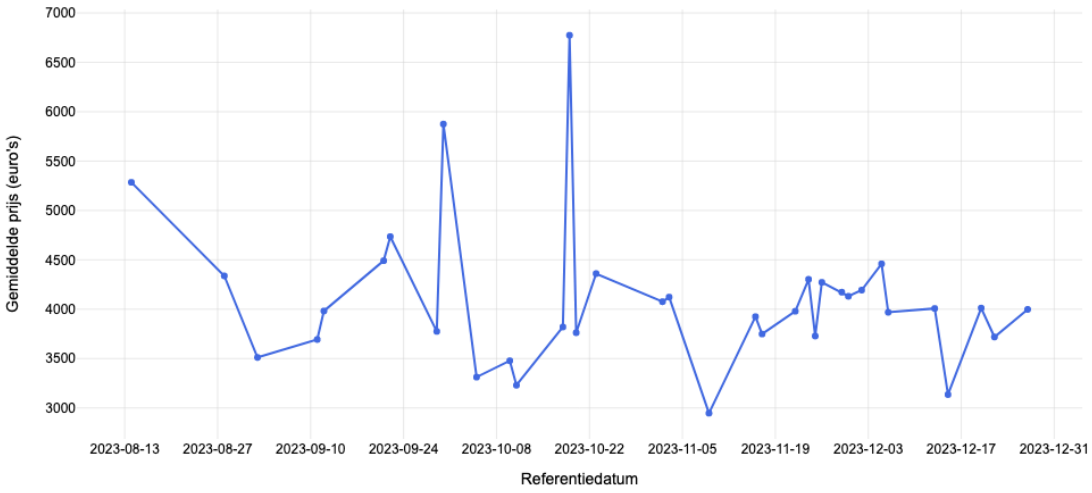
Figuur 2: Relatie tussen de prijs en het woonoppervlakte

Dit sterke verband tussen de vraagprijs en de woonoppervlakte wordt nogmaals bevestigd door het berekenen van de correlaties tussen de prijs en verschillende variabele. In Tabel 4zijn deze correlaties zichtbaar. Er is te zien dat de woonoppervlakte het sterkste verband met de prijs houdt. Daarna volgen het aantal kamers en de landoppervlakte.

Tabel 4: Correlatie features met vraagprijs

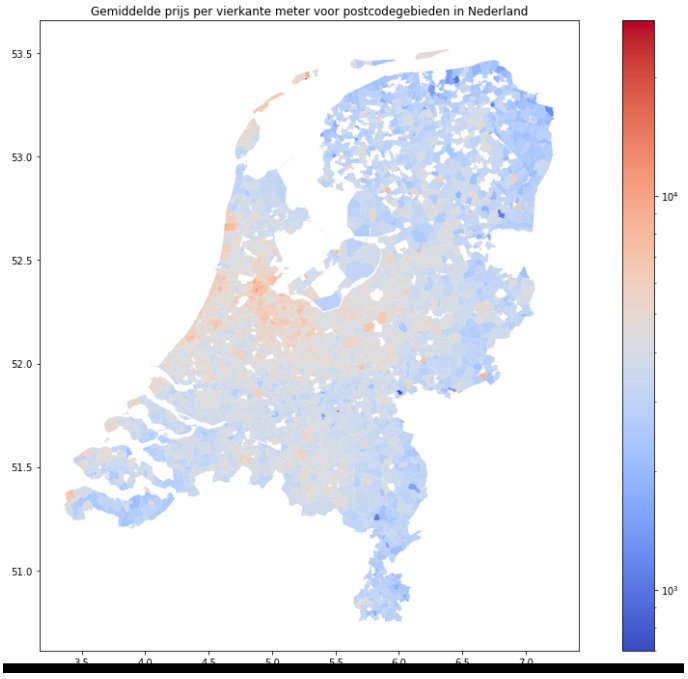
Feature	**Correlatie met vraagprijs**
living area	0.695301
number of rooms	0.441261
land area	0.289570
floor	0.211099
geodata latitude	0.042286
service charges	0.006290
construction year	-0.067617
geodata longitude	-0.082967
renovation year	-0.176725

Verder is de prijs tegenover de tijd uitgezet. Dit om te kijken of er in de drie maanden tijd verandering heeft plaatsgevonden in de prijzen. Dit is echter niet aan te tonen in figuur 3. Aangezien de tijd is geplot op basis van de aan ons aangeleverde datum in de dataset, komt deze voornamelijk overeen met de datum waarop de huizen in de dataset zijn geplaatst en niet de datum waarop de huizen op de verkoop websites zijn gezet. Hierdoor zou het kunnen zijn dat deze grafiek geen verband laat zien. Daarnaast blijken drie maanden dus ook te kort om echt iets over de verandering in de huizenmarkt te zeggen.



Figuur 3: Gemiddelde prijs per referentie moment

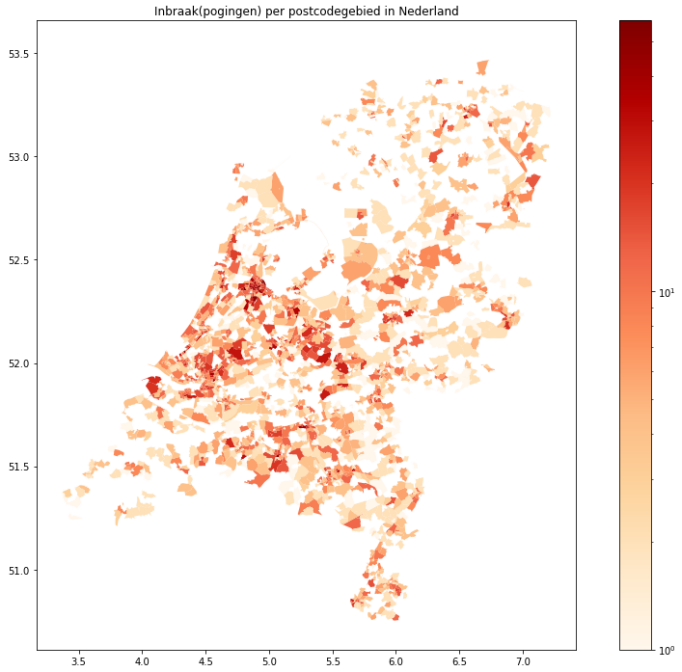
Als laatste is er een kaart gemaakt met de locaties en de vraagprijs per vierkante meter, gebaseerd op de postcodegebieden (zie figuur4). Er is gekozen voor een logaritmische schaalverdeling vanwege de aanwezigheid van uitschieters, dit maakt een duidelijkere visualisatie mogelijk. Uit dit figuur is op te maken dat de datapunten over heel Nederland verspreid liggen, enkel de witte punten zijn postcodegebieden zonder data. Figuur 4 toont ook dat de prijzen (per vierkante meter woonoppervlakte) het hoogste liggen in stedelijke gebieden, die piek is voornamelijk in de randstad te zien en met name Groningen, Leeuwarden en Limburg zijn goedkoper.



Figuur 4: Kaart met de vraagprijs per vierkante meter op basis van het postcodegebied

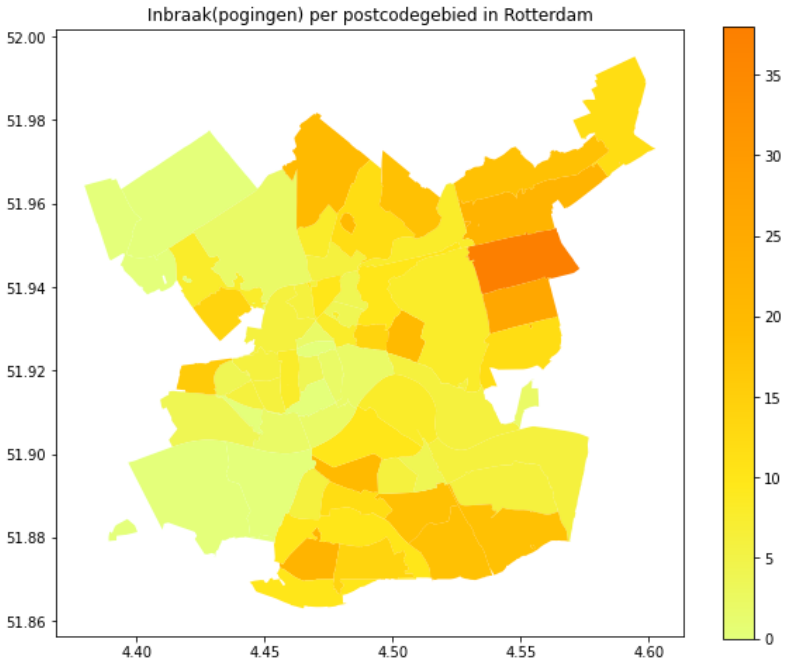
2.4 Prijzen en criminaliteit

Om een beeld te krijgen van hoe de inbraken verdeeld zijn over Nederland, is dit afgebeeld, zie 5. De witte gebieden zijn de postcode zones zonder inbraken in de afgelopen drie maanden. Uit de plot is op te maken dat voornamelijk in de randstad en andere stedelijke gebieden veel woninginbraken zijn. Dit komt overeen met de hogere prijzen in stedelijke gebieden zoals te zien in figuur 4. Het is logisch om hier een verband te zien omdat dit ook de gebieden zijn met de hoogste bevolkingsdichtheid (voor de Statistiek, 2019).

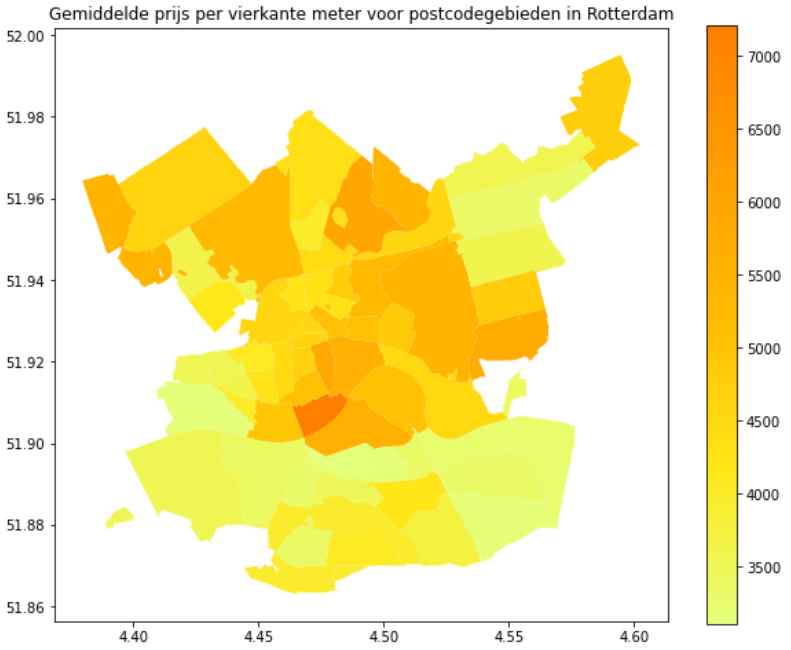


Figuur 5: Kaart met het aantal inbraken per postcodegebied

Uit de kaart hierboven lijkt er een verband te bestaan tussen de vraagprijs per vierkante meter en het aantal inbraken. Hier dient voornamelijk de bevolkingsdichtheid als een logische verklaring. Het is daarom ook interessant om binnen steden te kijken naar patronen. Hieronder is in figuur 7 de prijs per vierkante meter in de stad Rotterdam afgebeeld. Daaronder is het aantal inbraken per postcodegebied geplot (figuur 6). Het is zichtbaar dat het Noord-Oosten van de stad een hogere inbraak aantal heeft en daarbij ook lagere prijzen. In andere delen van de stad lijken lagere inbraak cijfers in combinatie te gaan met de hogere huizenprijzen. Dit verband is echter niet heel duidelijk zichtbaar en niet significant aan te tonen.



Figuur 6: Kaart met het aantal inbraken per postcodegebied



Figuur 7: Kaart met de prijs per postcodegebied

3 Benadering

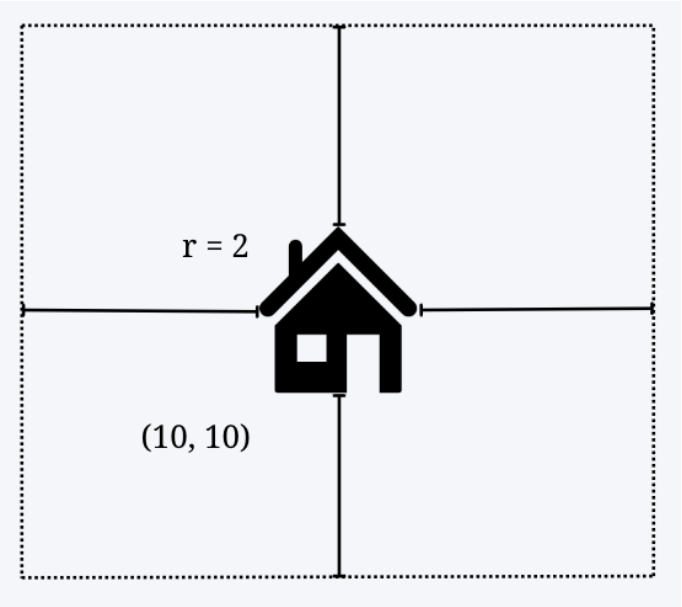
Zoals eerder toegelicht is het doel van dit onderzoek het creëren van een model dat aan de hand van (alternatieve data) een voorspelling kan maken van de waarde van een huis. Er zijn verschillende technieken overwogen gedurende dit onderzoek. Om de uiteindelijke keuze van de technieken te begrijpen, is het naast een goede grip op de data, van belang om technisch begrip te hebben van de toepassingen. In dit hoofdstuk wordt ten eerste kort de theoretische achtergrond van de modelleer technieken besproken, wordt er volgens ingegaan op de wijze van implementatie van het model en ten slotte wordt beschreven welke *features* en *hyperparameters* gebruikt worden en hoe de prestaties van de modellen worden gemeten.

3.1 Individuele modellen

K-nearest neighbours

K-nearest neighbours (KNN) is een eenvoudig en intuïtief machine learning-algoritme dat wordt gebruikt voor zowel classificatie als regressie. Het is gebaseerd op het idee dat soortgelijke datapunten neiging hebben om dicht bij elkaar te liggen in de feature-ruimte. De essentie van de techniek is het classificeren of voorspellen van een nieuw datapunt op basis van de k dichtstbij liggende punten in de trainingsdataset.

Bij het gebruik van KNN moet de gehele dataset in het geheugen worden geladen. Om de dichstbijzijnde buren te vinden van het querypunt moet immers de afstand tussen dit punt en alle andere punten worden berekend. Omdat dit moet worden berekend over ongeveer 155.000 datapunten, kost dit een enorme hoeveelheid geheugen en tijd. Er is hierom voor gekozen om een variatie van deze methode te gebruiken die meer op *"nearest neighbour"* lijkt (zie figuur **??**).



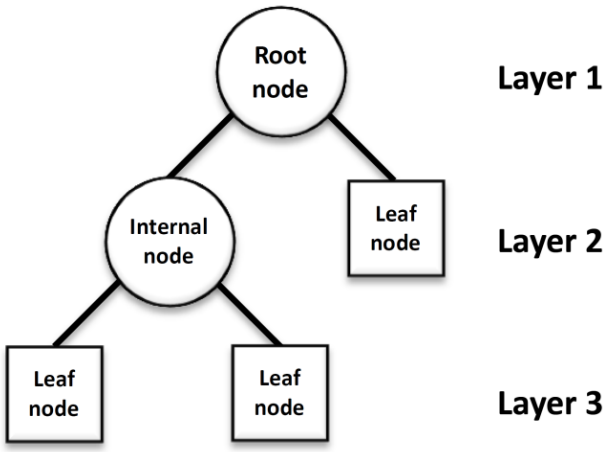
Figuur 8: Implementatie van nearest neighbour; box tekening

In deze implementatie wordt er niet gekeken naar het k aantal trainpunten het dichtst bij het testpunt, maar wordt er een box om het testpunt heen gemaakt en worden de datapunten die in die box zitten hieruit meegenomen. Om dit te versnellen wordt de data van tevoren gesorteerd op de coördinaten. Vervolgens wordt het gemiddelde genomen van deze datapunten. Dit representeert de gemiddelde prijs per vierkante meter in die buurt. Met een vermenigvuldiging met het woonoppervlak wordt dan de vraagprijs voorspeld. Belangrijk om te vermelden is wel dat het model hierdoor een bias heeft; in de praktijk is een huis dat twee keer zo groot is over het algemeen minder dan twee keer zo duur (zie ook figuur 2).

Decision tree

Een *decision tree* is een boomachtige modelstructuur die wordt gebruikt voor zowel classificatie als regressie, zie figuur 9. Het model maakt beslissingen op basis van kenmerken van de gegevens om tot een eindresultaat te komen. *Decision trees* zijn flexibel en kunnen goed omgaan met complexe relaties in data en geven een intuïtieve benadering bij het modelleren van beslissingsprocessen. Bovendien geven ze inzicht in welke features van de dataset belangrijk zijn. Deze staan aan de bovenkant van de boom, want hiermee worden de grootste splits in de dataset gedaan. Het besluitvormingsproces van een *decision tree* gaat als volgt:

Ten eerste wordt er een feature gekozen die voor de grootste afname aan 'onzekerheid' zorgt en daarmee voor de grootste informatiewinst. De data wordt daarbij opgesplitst in *subsets* op basis van de waarden van de eerstgekozen feature. Dit vormt de eerste splitsing in de *decision tree*, waardoor er een volgende laag aan zogeheten *nodes*ontstaat. Ditzelfde herhaalt zich voor deze *subsets* waardoor er zich een boomstructuur met takken en subtakken ontwikkelt. Als er geen informatiewinst meer te behalen valt, eindigt de boomstructuur in een *leaf node*; het eindpunt waar een definitieve voorspelling wordt gedaan van de *target variable*.



Figuur 9: Visualisatie van *decision tree*

In de functie voor de *decision tree* wordt de `DecisionTreeRegressor` *library* van `sklearn` gebruikt. Aan deze functie kunnen een aantal aspecten worden gespecificeerd en aangepast zoals hoe de boom gesplit moet worden.

Lineaire regressie

Dit statistische model model kan de relatie tussen een afhankelijke variabele (target) en één of meer onafhankelijke variabelen (features) modelleren. Het doel is om een lineaire relatie te vinden die het beste de variabiliteit in de afhankelijke variabele verklaart. Er zijn een tweetal lineaire regressies uitgevoerd, geïmplementeerd als volgt:

```
 Ridge:  from sklearn.linear_model import Ridge
Lasso:  from sklearn.linear_model import Lasso
```

Een *polynomial regression* was mogelijk met behulp van `from sklearn.preprocessing` import *PolynomialFeatures*

MLP Regressor

Een *MLP regressor* is een een type model dat onder de neurale netwerken is in te delen. Het bestaat uit een netwerk van kunstmatige neuronen, georganiseerd in lagen. De inputlaag ontvangt de inputkenmerken van het probleem, terwijl de outputlaag de voorspelde continue uitvoer produceert. Tussen de input- en outpulagen kunnen één of meerdere verborgen lagen voorkomen, waarin neuronen informatie verwerken via lineaire en niet-lineaire transformaties.

Elke neuron in het netwerk past een activatiefunctie toe op de gewogen som van zijn input om op deze manier de output te berekenen. Dit wordt gedaan door de invoer van het neuron te vermenigvuldigen met de gewichten van de verbindingen en deze resultaten bij elkaar op te tellen, waarna een activatiefunctie wordt toegepast op het resulterende totaal. Soms past men hier een bias op toe. Veelgebruikte activatiefuncties zijn de sigmoid-functie, de hyperbolische tangent-functie (tanh) en de ReLU-functie (Rectified Linear Unit).

De MLP Regressor wordt getraind door de gewichten en biases op zo'n manier aan te passen dat de output zo dicht mogelijk bij de daadwerkelijke targetwaarde terechtkomt. Dit wordt meestal gedaan door middel van optimalisatie-algoritmen zoals backpropagation, waarbij de afgeleiden van een foutfunctie met betrekking tot de gewichten en biases worden gebruikt om de gewichten en biases geleidelijk aan te passen in de richting die de fout vermindert.

De Regressor is met behulp van de volgende library in de code verwerkt:

`MLPRegressor: from sklearn.neural_network import MLPRegressor`

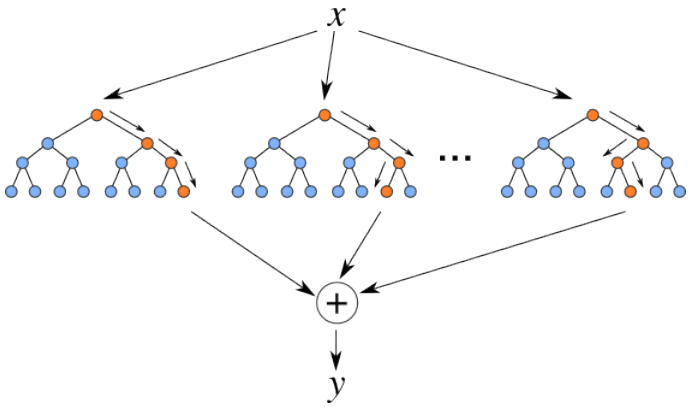
3.2 Ensemble modellen

Ensemble methoden vertegenwoordigen een krachtige benadering binnen ML. De methoden maken gebruik van de aggregatie van meerdere individuele modellen en profiteren van de diverse sterke punten van elk individueel model. Deze aggregatie zorgt over het algemeen voor betere voorspellende prestaties en meer robuustheid in vergelijking met individuele modellen; de zwakke punten van de individuele modellen worden gecompenseerd. Er zijn een viertal type ensemble modellen te onderscheiden: *Bagging*, *Boosting*, *Stacking* en *Voting*. Omdat Bagging en Boosting het best bestendig (bleken te) zijn tegen *overfitting*, zijn deze twee methoden uiteindelijk geïmplementeerd.

Bagging

Bagging is een ensemble methode waarbij meerdere exemplaren van een enkelvoudig ML-algoritme worden getraind op verschillende subsets van de trainingsdata. Deze subsets worden gegenereerd door middel van *bootstrap sampling*, waarbij willekeurige monsters met vervanging worden genomen uit de oorspronkelijke dataset. Vervolgens worden de voorspellingen van elk model gemiddeld om een uiteindelijke voorspelling te verkrijgen. Voorbeelden van bagging algoritmen zijn *Random Forests* en een *Bagging Regressor*. Deze beide algoritmen zijn toegepast in dit onderzoek.

In plaats van één beslissingsboom, wordt er bij zowel de *Random Forest*-methode als bij de *Bagging Regressor* een ensemble of een "bos"van vele beslissingsbomen gecreëerd. Voor elk van de beslissingsbomen wordt een willekeurig subsample van de traingsdata gemaakt door *bootstrapping* toe te passen. *Bootstrapping* houdt in dat voor elk van de bomen een nieuwe trainingsdataset wordt gemaakt door willekeurig samples met vervanging uit de oorspronkelijke dataset te trekken; bij het trekken van elk sample voor een nieuwe steekproef, wordt diezelfde sample teruggeplaatst in de dataset voordat het volgende sample wordt getrokken. Op deze manier kan elke sample meerdere keren worden opgenomen in dezelfde steekproef en kan het ook zo zijn dat sommige voorbeelden geheel niet worden meegenomen. Zo worden er telkens nieuwe steekproeven gegenereerd die vergelijkbaar zijn met de oorspronkelijke dataset, maar dan net met kleine variaties. Deze nieuwe steekproeven worden vervolgens gebruikt om de meerdere *decision trees* te trainen. Zie figuur 10:



Figuur 10: Visualisatie van Random Forest

Vanaf dit punt gaan de *Random Forest* en de *Bagging Regressor* van elkaar verschillen. Bij de *Bagging Regressor* wordt bij elke mogelijke splitsing álle features overwogen op splitsing, terwijl bij *Random Forest* enkel een willekeurige subset van de features meegenomen. Dit zorgt voor een grotere diversiteit tussen de bomen en kan de voorspellende prestaties verbeteren door overfitting te verminderen en robuustere modellen te produceren.

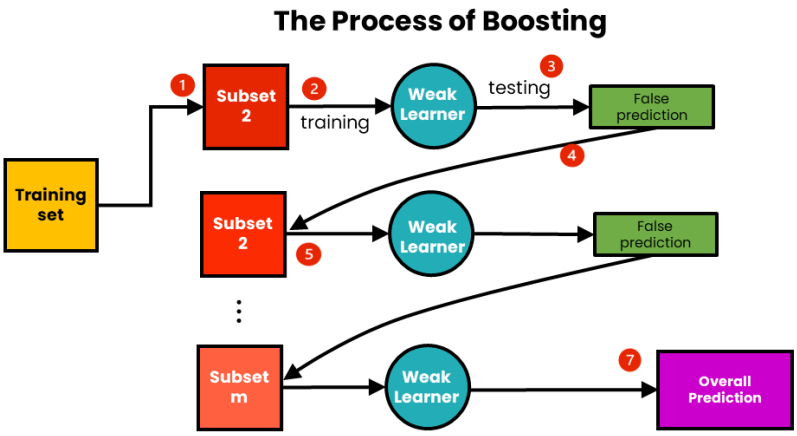
De belangrijkste voordelen van *Random Forest* en de *Bagging Regressor* zijn dat ze zeer robuust zijn en goed presteren op een breed scala aan taken, zonder dat daarvoor veel parameters moet worden afgesteld. Ook zijn ze robuuster en beter bestand tegen overfitting dan *decision trees* an sich.

De bagging methoden zijn met de volgende libraries verwerkt:

```
   Random forest:  from sklearn.ensemble import RandomForestRegressor
Bagging Regressor:  from sklearn.ensemble import BaggingRegressor
```

Boosting

Boosting is een ensemble methode die zich richt op het sequentieel trainen van zwakke modellen om de algehele voorspellende prestaties te verbeteren. In tegenstelling tot *bagging*, waarbij elk model los van elkaar wordt getraind, richt *boosting* zich op het corrigeren van de fouten van eerdere modellen door als het ware meer 'gewicht' te geven aan foutief voorspelde waarden in de trainingsdata (zie figuur 11 Voorbeelden van dit type algoritmen zijn *Gradient Boosting* en *XGBoost*.



Figuur 11: Voorbeeld werking Boosting algoritme

XGBoost (eXtreme Gradient Boosting) is een geavanceerde versie van *Gradient Boosting* om de prestaties van traditionele methode te verbeteren. Zo voegt XGBoost onder anderen L1- en L2-regularisatie toe waardoor overfitting wordt voorkomen en de voorspellingen nauwkeuriger worden.

De Boosting algoritmen zijn als volgt verwerkt:

```
GradientBoosting:  from sklearn.ensemble import GradientBoostingRegressor
XGBoost:  from xgboost import XGBRegressor
```

3.3 Implementatie van de modellen

Er is één grote functie gemaakt waarbij gekozen kan worden voor de volgende opties:

1. De te gebruiken data
2. Het te gebruiken model
3. De gewenste features
4. Welke k voor *k-fold cross-validation*
5. De hoeveelheid jobs voor parallelisatie
6. Of de voorspelling moet worden opgeslagen in een DataFrame, csv of niet
7. Of de resultaten geprint moeten worden, als output moeten worden gegeven of beiden
8. Of de hoeveelheid verwijderde datapunten moeten worden geprint (in het geval er NaN-waarden zijn gevonden voor de gebruikte features)
9. Of de hoeveelheid gebruikte datapunten in het model geprint moet worden
10. De optie om alle datapunten met NaN-waarden van features te verwijderen (ongeacht of deze features worden meegenomen of niet)

3.4 *Model performance measures*

3.4.1 *K-fold cross-validation*

Er is gebruik gemaakt van *k-fold cross validation*, waardoor de stabiliteit van de resultaten hoger is dan als er enkele train- en testset zouden zijn gemaakt. Bij *k-fold cross validation* wordt de gehele dataset verdeeld in k gelijke subsets, waarbij de dataset van tevoren wordt geschud. Hierna wordt elk van deze subsets één keer als testset gebruikt, terwijl alle andere subsets dan

samen als trainset dienen. Zo zit elk punt een keer in de testset en kan dus voor elk punt een voorspelling worden gemaakt. Voor elke testset worden vervolgens de errors berekend en deze worden gemiddeld om de prestaties van het hele model te meten.

Een mogelijkheid is om k gelijk te maken aan het aantal datapunten in de dataset. In dat geval wordt elke prijs voorspeld op basis van alle andere datapunten. Dit wordt *leave-one-out cross-validation* genoemd. Voordelen hiervan zijn dat de exacte voorspellingen gebruikt worden op de manier zoals die ook voor een enkel nieuw punt berekend worden en dat deze voorspellingen niet veranderen als deze opnieuw worden berekend met dezelfde dataset (waar het met andere waardes voor k wel iets zou veranderen omdat de verdeling van de punten in de subsets zou veranderen). Het nadeel hieraan is dat het uitvoeren hiervan veel tijd vergt.

3.4.2 *Execution time*

Er wordt gebruik gemaakt van de *time library* in *Python*. Voordat elk model wordt uitgevoerd wordt de tijd vastgezet in een variabele en op het moment dat het model klaar is, wordt de tijd weer in een variabele gezet. Dan wordt de eindtijd min de starttijd gedaan en dan is bekend hoe lang het model nodig gehad heeft. In het hoofdstuk snelheidsoptimalisatie wordt behandeld op welke manier *hyper parameters* kunnen aangepast worden om betere nauwkeurigheid te behalen als blijkt dat het model heel snel is, en aan de andere kant zou nauwkeurigheid opgeofferd kunnen worden om de snelheid van het model te verbeteren.

3.4.3 Bias percentage

De bias geeft weer in welke mate het model de prijzen over- of onderschat. Vaak kan het model verbeterd worden door de bias dichter bij nul te brengen. De berekening is
$\sum_{i=1}^{N}(\frac{(voorspelling_i - prijs_i)*100}{prijs_i})/N.$

3.4.4 *Mean absolute error percentage*

Dit is het gemiddelde percentage dat de voorspelling gemiddeld genomen van de prijs afligt. Hoe lager, hoe beter. De berekening is
$\sum_{i=1}^{N}(\frac{|voorspelling_i - prijs_i|*100}{prijs_i})/N.$

3.4.5 *Root mean squared error percentage*

Deze waarde is minder intuïtief, maar het zorgt ervoor dat voorspellingen die ver van de echte prijs afliggen nog erger zijn, doordat het verschil in het kwadraad gedaan wordt. Hoe lager hoe beter. De berekening is
$\sqrt{(\sum_{i=1}^{N}(\frac{(voorspelling_i - prijs_i)^2*100}{prijs_i})/N)}.$

3.4.6 R^2 *score*

Dit is de determinatie coëfficiënt. Om deze te berekenen wordt *from sklearn.metrics import r2_score* gebruikt. Dit geeft de relatie tussen twee variabelen aan. In dit geval zijn de twee variabelen de voorspelde prijzen en de echte prijzen. Hoe hoger, hoe beter (max is 1).

3.5 *Feature selection*

Het bepalen welke variabelen meegenomen worden in ieder model is een belangrijke taak, omdat de resultaten hierdoor sterk kunnen veranderen. Vaak kan bijvoorbeeld de correlatiematrix tussen variabelen (waaronder de prijs) worden gebruikt. In dit geval worden alle datapunten verwijderd waarvoor er een NaN-waarde bestaat voor de variabele die gebruikt wordt. Daarom kan het zo zijn dat een variabele wel een goede correlatie heeft met de prijs, maar het de modellen toch slechter maakt, doordat de trainingsset kleiner wordt. Er bestaat daarnaast de optie om de voorspellingen van andere modellen te gebruiken als variabele in een ander model. Belangrijk om te vermelden is wel dat er op die manier geen voorspelling is voor datapunten die een NaN-waarde

hadden voor een variabele die in het vorige model is gebruikt. Daarom is er gekozen om honderden mogelijke combinaties van variabelen en modellen te gebruiken om te zien welke combinatie de errors van het beste model zo laag mogelijk kregen. In de gemaakte functie is het gemakkelijk om de gebruikte variabelen voor elk model aan te passen. In dit onderzoek is dat handmatig gedaan. De combinatie van variabelen voor elk model dat is gebruikt is als volgt:

	living_area	**land_area**	**const_year**	**renovation**	**energy**	**rooms**	**prop_type**	**NN**
NN	X							-
Rd	X	X	X			X		X
DT	X	X	X		X	X		X
MLP	X	X		X	X	X	X	
RF	X	X	X		X	X		X
GB		X			X		X	X
XGB		X			X			X
BR	X	X	X	X	X	X	X	X
SVR	X	X	X	X	X	X	X	X
Ls	X	X	X	X	X	X	X	X
AB	X	X	X	X	X	X	X	X
NM	X	X	X	X	X	X	X	X

Tabel 5: De gebruikte *features* in elk model, aangeduid met 'X'

NN = Nearest Neighbours Regressor
Rd = Ridge Regressor
DT = Decision Tree Regressor
MLP = Multilayer Perceptron Regressor
RF = Random Forest Regressor
GB = Gradient Boosting Regressor
XGB = XGBoosting Regressor
BR = Bagging Regressor
SVR = Support Vector Regressor
Ls = Lasso Regressor
AB = AdaBoosting Regressor
NM = Neural Model (keras)

3.6 *Hyperparameter optimalisation*

De *hyperparameters* zijn de argumenten van modellen waarmee de snelheid en nauwkeurigheid (en de balans hiertussen) geoptimaliseerd kunnen worden. Hoe hoger de k (voor *k-fold*), hoe langer het duurt, maar wel hoe betrouwbaarder de resultaten worden. Er is gekozen voor een k=100 voor Rotterdam en een k=15 voor het hele land. De radius van *Nearest Neighbours* is bepaald door veel waarden te proberen in een loop. Er is gesteld dat de verhouding tussen het aantal huizen zonder en met voorspelling maximaal 0,0001 mag zijn (dit heet een *constraint*). Onder die voorwaarde is de radius met het laagste *Mean Absolute Error Percentage* genomen. De radius die eruit kwam gebaseerd op het hele land is 1,2 km. Per stad kan er een andere optimale waarde zijn.

De *alpha* voor *Ridge* en *Lasso* regression is op 1 gezet. Dit is de standaard waarde en de verschillen in resultaten met andere waarden is minimaal. De beste polynomiale graad is 2. Hiervoor zijn alle errors het laagst. Voor sommige steden kan 1 of 3 beter zijn.

Het gebruikte *criterion* om de *Decision Tree* en *Random Forest* op te splitten is *friedman-mse*. Dit bleek de laagste errors te geven van alle opties.

De *Multilayer Perceptron* gebruikt een enkele *hidden layer* met 100 neuronen. Meer lagen gaven geen verbeterde resultaten en maakte het model trager. Minder dan 80 en meer dan 120 neuronen zorgde voor een verhoging in de errors. Er is daarom gekozen voor een gebalanceerde 100. De activatiefunctie voor de *hidden layer* is *relu*. De activatiefunctie voor de *output layer* is de *identity* functie. *max_iter* is 2000. Dit is tien keer de standaardwaarde en het model is nog steeds redelijk snel. De *solver* is *adam*. *lbfgs* gaf iets betere resultaten, maar er kwamen foutmeldingen over dat de *max_iter* werd overschreden; zelfs als deze boven de 10000 stond. *batch size* staat op automatisch.

Bagging Regressor gebruikt *XGBoost* als model om te *baggen*. Dit met onderstaande *hyperparameters*: *XGBoost* gebruikt $n\_estimators = 100$ en $max\_depth = 8$. Er is hiervoor gekozen omdat de *Bagging Regressor* vrij traag wordt als deze waarden verhoogd worden. Daarnaast is er weinig verschil terug te zien in de resultaten.

Er is voor gekozen om de *Voting Regressor*, de *Stacking Regressor* en het *keras* neurale netwerk niet te gebruiken vanwege minder goede resultaten de resultaten en trage werking. Tenslotte zijn van alle ongenoemde *hyperparameters* de standaard waarden van de gebruikte *libraries* gebruikt.

3.7 Snelheidsoptimalisatie

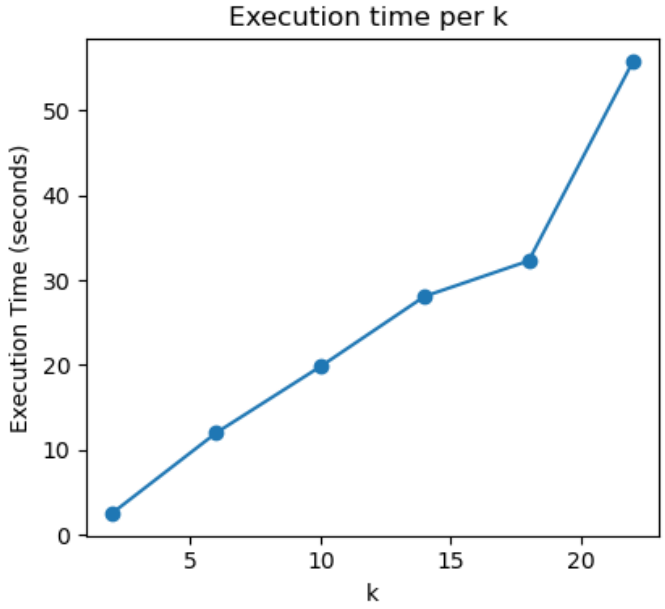
3.7.1 *Parallel processing*

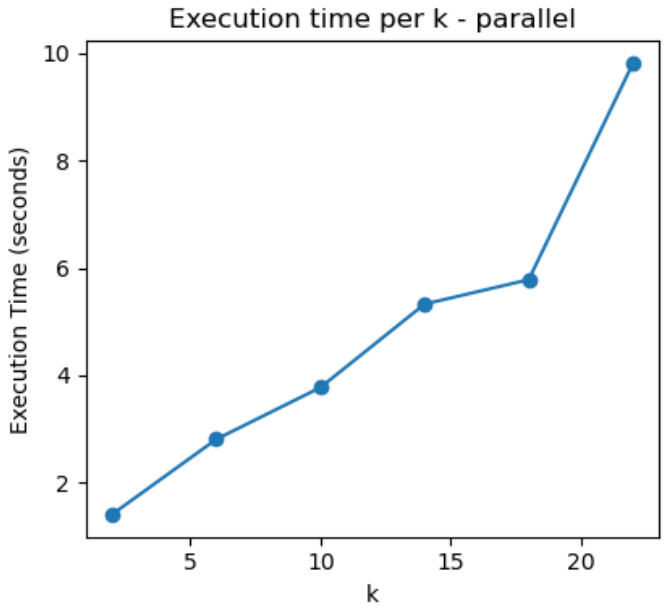
Om de snelheid te verbeteren zijn de `Parallel en delayed libraries` van `joblib` gebruikt. Dit zorgt ervoor dat processen parallel uitgevoerd kunnen worden op de CPU. Dit werkt goed in combinatie met *k-fold cross-validation*, want de berekeningen voor elke subset kunnen tegelijk worden uitgevoerd. Hierna worden de uitslagen even opgeslagen totdat alle subsets klaar zijn en dan worden ze samengevoegd in een *ndarray*. Het aantal processen dat tegelijk uitgevoerd wordt, wordt aangeduid als *n_jobs*. De beste waarde verschilt per CPU, maar door -1 in te vullen, wordt automatisch het maximaal aantal processen gebruikt. Dit maakt de modellen bijna altijd veel sneller, maar in gevallen dat de dataset heel klein is of de k heel laag is, duurt het maken van parallele processen soms langer dan het los berekenen van alle resultaten zou zijn.

Met hele grote datasets kan het ook zo zijn dat er problemen ontstaan omtrent werkgeheugen. Als voor elk proces de hele dataset ingeladen moet worden en dit meerdere keren tegelijk moet gebeuren, kan het aantal *bytes* dat moet worden onthouden groter worden dan het beschikbare werkgeheugen en dan loopt het programma vast en wordt dus slomer. Een ander nadeel is dat als het uitvoeren onderbroken wordt, het programma vastloopt. Hierom is besloten om ook en optie toe te voegen om de resultaten te berekenen zonder het gebruik van *parallel processing*. Ook gebruiken sommige modellen uit *libraries* automatisch al *parallel processing* of worden het model uitgevoerd op de GPU. In die gevallen wordt de snelheid vaak niet beter.

Hieronder is het verschil in snelheid te zien tussen het parallel uitvoeren met n_jobs = -1 en het niet parallel uitvoeren van de *Multilayer Perceptron Regressor* (met de hyperparameters die onder 4.5.2 beschreven worden) voor k = 2, 6, 10, 14, 18 en 22. Dit werd enkel gedaan voor de huizen die in Rotterdam liggen, omdat het anders erg lang zou duren.

Te zien is dat parallel hier altijd sneller is. Opvallend is de grote sprong in tijd van 18 naar 22. Dit komt doordat er een limiet is aan hoeveel processen er tegelijk kunnen worden uitgevoerd totdat het werkgeheugen vol zit, of de CPU geen plaats meer overheeft voor meer processen. De waarde waarvoor dit gebeurt is afhankelijk van de CPU die gebruikt wordt en kan ook veranderen door het aantal processen dat op de achtergrond bezig is.





3.7.2 Uitvoeren op de GPU

Als er veel rekensommen achter elkaar uitgevoerd moeten worden, kan het veel sneller zijn om deze uit te voeren met behulp van de GPU in plaats van de CPU. De CPU is echter beter als er veel in het geheugen verplaatst moet worden. Een goed voorbeeld van een model dat erg geschikt is voor de GPU is het neurale netwerk. Het neurale netwerk van *keras* is speciaal gemaakt om via de GPU uit te voeren. Dit is ook gedaan met behulp van *Tensorflow*, *CUDA* en *cuDNN*. Dit is gedaan via *Windows* 11, dus moesten er oude versies gebruikt worden, want GPU ondersteuning voor *Windows* is er niet meer voor *Tensorflow* sinds versie 2.11. De *CUDA* en *cuDNN* versies moesten daar ook op worden aangepast. De gebruikte GPU is een *NVIDIA RTX 3060 mobile*. Er is goed gecontrolleerd of het echt op de GPU werd uitgevoerd (en dat was het geval), maar toch was dit neurale netwerk een stuk minder snel dan de *Multilayer Perceptron* van *scikit learn* die op de CPU uitgevoerd werd. Aangezien de resultaten erg vergelijkbaar waren, is er gekozen om alleen de *Multilayer Perceptron* van *scikit learn* te optimaliseren voor het hele land.

De snelheden voor de verschillende modellen voor Rotterdam met $k = 5$ zijn als volgt (hieruit wordt ook duidelijk waarom drie modellen niet gebruikt worden):

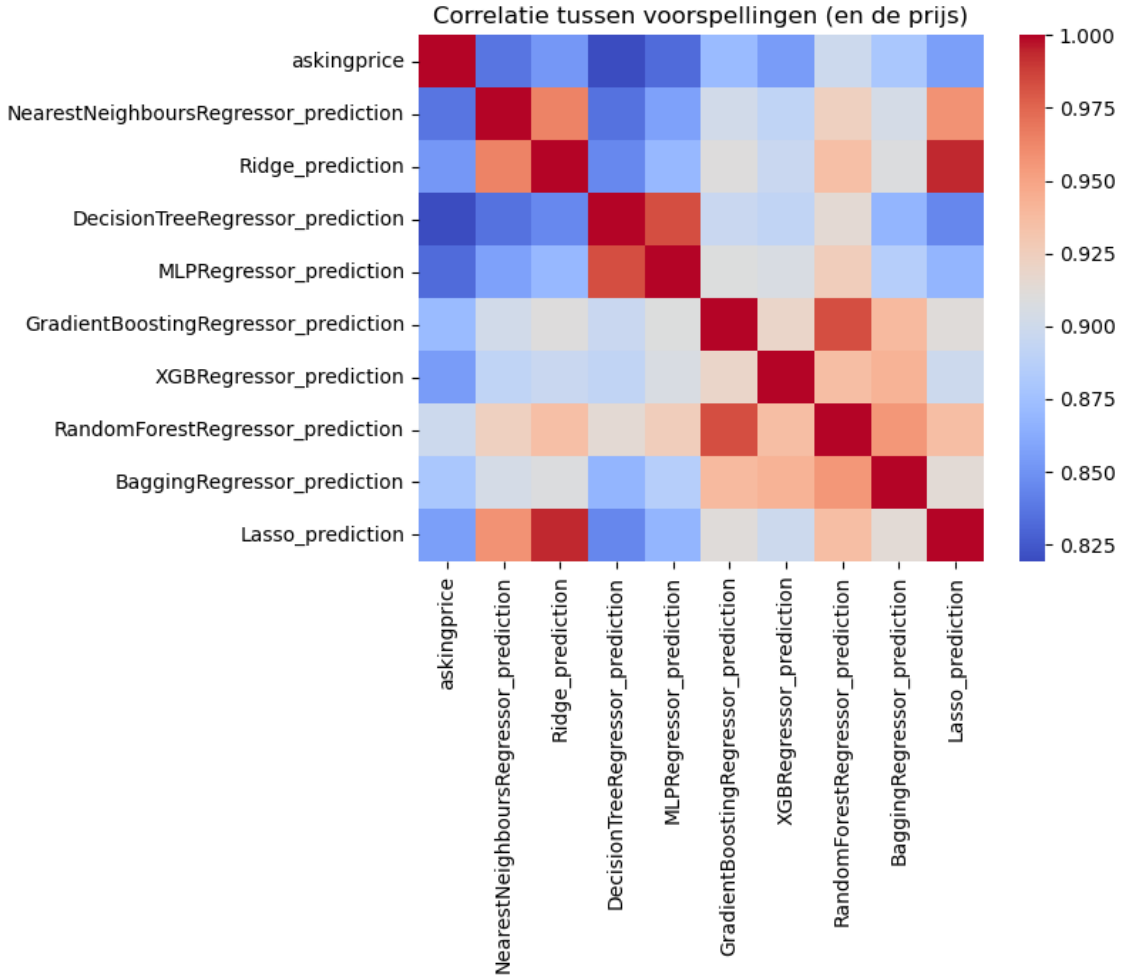
Model	**Execution Time (seconden)**
Lasso	0.291256
XGBRegressor	0.309867
AdaBoostRegressor	0.395568
GradientBoostingRegressor	0.448061
SVR	0.632189
RandomForestRegressor	1.189312
MLPRegressor	1.301595
DecisionTreeRegressor	1.385674
Ridge	1.463223
BaggingRegressor	1.749277
NearestNeighboursRegressor	4.373729
VotingRegressor	3.908138
StackingRegressor	19.365350
Sequential	106.510738

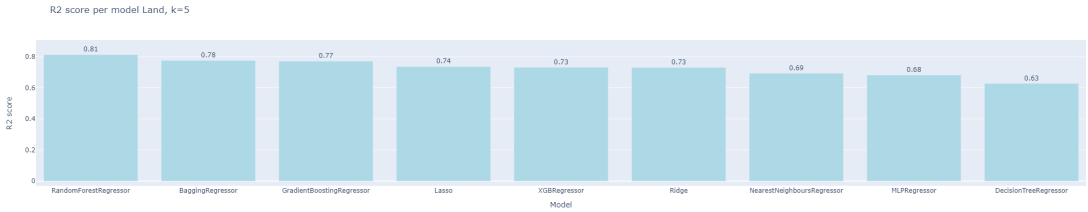
Tabel 6: *Execution Time* per model

4 Resultaten

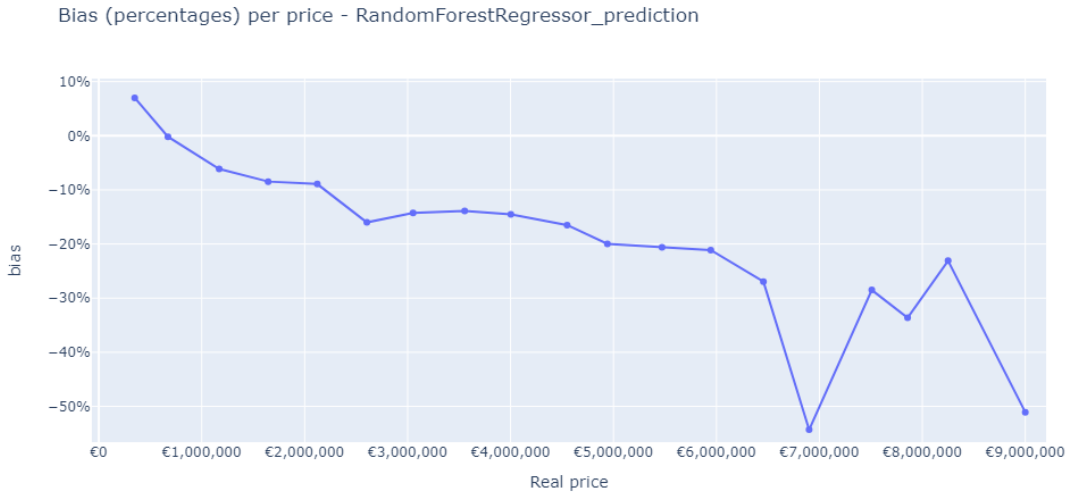
4.1 Voorspellen van huizenprijzen

Covarience tussen modellen - land, k=5

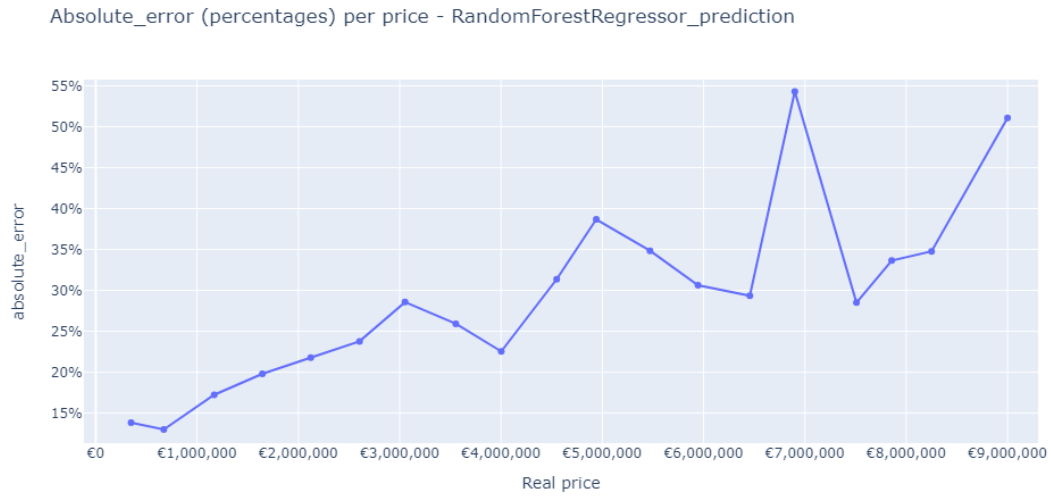


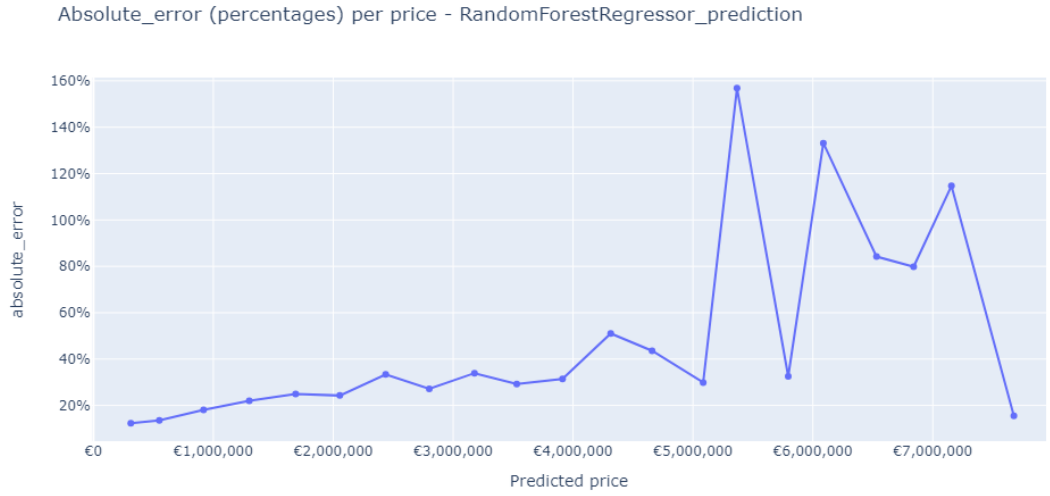


Bias per prijs - land, k=5



MAE per prijs - land, k=5





4.2 Invloed van inbraakcijfers op voorspellingen

Van tevoren zijn alle datapunten zonder postcode verwijderd; zowel bij de berekeningen met als bij de berekeningen zonder inbraakcijfers. Hier is voor gekozen om ervoor te zorgen dat alleen de invloed van de alternatieve data te zien is en dit dus niet te verklaren is door missende data.

Tabel 7: Resultaten Nederland - koop - zonder inbraak - *k=15*

	Ridge	DT	MLP	RF	GB	XGB	Bagging
Bias	4.1539%	3.8833%	**0.2229%**	3.9886%	4.1123%	4.1768%	3.8128%
MAE	17.1627%	19.2964%	19.0045%	14.3678%	14.6145%	16.0212%	**14.1421%**
RMSE	25.6273%	30.362%	28.8632%	23.0358%	23.5354%	26.2773%	**22.8378%**
Mean R^2	0.7524	0.6055	0.6569	**0.7808**	0.7471	0.6733	0.7797

Tabel 8: Resultaten Nederland - koop - met inbraak - *k=15*

	Ridge	DT	MLP	RF	GB	XGB	Bagging
Bias	4.0811%	3.7689%	**-3.3532%**	3.9358%	4.1379%	4.0816%	3.9762%
MAE	17.6225%	19.656%	19.6421%	14.6048%	14.83%	15.8312%	**14.3858%**
RMSE	26.5175%	30.1828%	30.7431%	24.7473%	**23.7001%**	25.1718%	23.8231%
Mean R^2	**0.793**	0.5911	0.6419	0.7859	0.788	0.5375	0.623

Tabel 9: Resultaten Rotterdam - koop - zonder inbraak - *k=15*

	Ridge	DT	MLP	RF	GB	XGB	Bagging
Bias	**1.6824%**	1.9458%	1.7166%	2.2222%	2.363%	2.245%	2.4739%
MAE	18.2875%	12.6141%	14.4527%	**10.7926%**	11.268%	12.3971%	11.0808%
RMSE	25.4456%	20.7528%	21.6503%	**16.4029%**	16.5745%	18.9175%	16.525%
Mean R^2	0.7031	0.6979	0.715	0.8176	**0.8323**	0.7877	0.8317

Tabel 10: Resultaten Rotterdam - koop - met inbraak - *k=15*

	Ridge	DT	MLP	RF	GB	XGB	Bagging
Bias	1.6113%	1.9704%	**1.3264%**	2.1988%	2.2392%	2.3617%	2.2905%
MAE	18.0627%	12.5576%	13.6408%	**10.5866%**	11.0086%	11.991%	10.7152%
RMSE	25.1341%	20.5642%	20.7937%	16.3588%	16.2167%	18.0684%	**16.1899%**
Mean R^2	0.6944	0.7322	0.7617	0.8177	**0.8451**	0.808	0.833

5 Conclusie

Gedurende dit project werd de impact van veiligheidsfactoren op de vastgoedmarkt onderzocht met behulp van criminaliteitsdata. Deze data, bestaande uit inbraakcijfers per postcodegebied, werden geïntegreerd in meerdere voorspellingsmodellen. Er werd hierbij gekeken naar de uiteindelijke invloed van het toevoegen aan deze data op de nauwkeurigheid van de voorspellingen van de modellen.

Als de modellen een voorspelling van de prijs van een huis doen, gegeven de data over heel Nederland, is er geen aantoonbare verbetering in resultaten te zien. Als echter de data wordt gefilterd op stad en de modellen worden gevraagd een voorspelling van de prijzen binnen deze stad te maken, is er wel aantoonbare verbetering te zien. Deze verbetering bedraagt zo voor de stad Rotterdam gemiddeld 0.10 % afname in Bias, 0.33% afname in de MAE, een afname van 1.06% in de RMSE en een toename van 0.02% in de Mean R score. Het Random Forest model bleek hierin de beste resultaten te behalen. Op basis van de resultaten van de modellen kan men dus stellen dat het betrekken van criminaliteitsdata bij het voorspellen van huizenprijzen inderdaad van toegevoegde waarde kan zijn, als men deze voorspellingen doet op stadsniveau.

6 Discussie

6.1 Discussie over te resultaten

6.1.1 Oorzaak en gevolg

Er kunnen vraagtekens worden gezet bij de stelling dat inbraak cijfers de prijzen van huizen beïnvloeden. Het is niet helemaal duidelijk welke kant het causatie verband op gaat, of dat beiden nog een andere, verborgen oorzaak hebben. Bovendien kan er door de inbrekers worden geredeneerd, bijvoorbeeld of er in de goedkoopste huizen wel iets te halen valt. Uiteindelijk is dit meer een sociologische kwestie. Voor het bepalen van een verkoopprijs op alternatieve data kan het feit dat er een verband is al interessant genoeg zijn om aan de modellen toe te voegen.

6.1.2 Tekortkomingen data

De modellen kunnen ten eerste verbeterd worden door de data te verbeteren. Veel van de missende waarden konden in dit onderzoek worden genegeerd, maar de waarden over vraagprijs, coördinaten en postcode waren essentieel, de laatste omdat de criminaliteitscijfers in postcodegebieden waren gegroepeerd. Vooral bij de postcodes misten er veel waarden, waardoor er 31019 datapunten uit de dataset met koopwoningen moesten worden gehaald en de modellen na deze opschoning minder goed werkten, omdat er minder data was om op te trainen.

Ten tweede kan de data worden uitgebreid door data te verzamelen over meerdere jaren. Criminaliteitscijfers worden meestal ook per jaar uitgegeven. Het is dus niet mogelijk om met huizenprijzen van enkele maanden voorspellingen te doen over de toekomst. Ook zou er beter inzicht over causatie kunnen worden gevonden, die uitbreidt op het in dit onderzoek gevonden verband.

Als laatste kan het een tekortkoming zijn dat de alternatieve data (de inbraak cijfers) maar van drie maanden tijd komen. Een inbraakgolf in een bepaalde buurt kan hier dus al best veel verschil maken. Dit gaat ten kosten van de generaliseerbaarheid van het model.

6.1.3 Enkele duplicaten overgebleven

Er is geprobeerd om van alle huizen die dubbel in de dataset stonden, de duplicaten te verwijderen, maar in een aantal gevallen stond een huis op meerdere websites tegelijk en die gevallen zijn er niet allemaal uitgehaald. Hiervoor zou er gekeken moeten worden naar het adres of naar de coördinaten. Echter hadden niet alle datapunten een adres met huisnummer en huizen die boven elkaar staan (appartementencomplex) hebben dezelfde coördinaten. Voor meer betrouwbare resultaten moeten deze gevallen er echter wel verwijdert worden (handmatig als het niet anders kan).

6.1.4 Verder onderzoek

Er bestaat een kloof tussen de beleefde veiligheid in een gebied en de werkelijke criminaliteit (Vallejo Velazquez, Kounadi & Podor, 2020). Het is mogelijk dat mensen die in de markt zijn voor een huis eerder afgaan van hun gevoel bij een buurt dan van criminaliteitsdata. In de toekomst kan worden onderzocht of er een sterker verband is tussen de huizenprijzen en de veiligheidsperceptie dan die tussen de prijzen en de criminaliteitsdata.

6.1.5 Vraagprijs is niet de verkoopprijs

Binnen dit onderzoek is de vraagprijs (`askingprice`) de *target variable*, dit is de variabele die wordt voorspeld. Wat belangrijk is om op te merken is dat het in de dataset gaat om de prijs die gevraagd wordt voor een woning. Dit is echter niet de prijs waarvoor het huis ook echt verkocht is. Op de huidige huizenmarkt is het vaak nodig om over te bieden op de gevraagde prijs (de Ondernemer, 2023).

6.1.6 Model zegt niet alles

Als een huis minder kost dan het volgens het model zou moeten kosten, betekent dat niet altijd dat het een goede deal is. De nauwkeurigheid is niet 100% en er wordt niet gekeken naar andere dingen dan de vraagprijs, zoals mogelijke *services* die inbegrepen zijn, makelaarskosten of dingen in de kleine lettertjes. Er moet dus altijd nog door een mens naar gekeken worden.

6.1.7 Opknappertje of nieuwbouw

Dit model zegt niks over de staat van een huis en of het geschikt is om meteen door te verkopen/verhuren of dat het huis eerst helemaal opgeknapt moet worden voordat er winst uitgehaald kan worden. Daar is een ander model voor nodig.

6.1.8 Spijtmodel en *compromise effect*

Mensen denken in de praktijk niet zoals deze modellen. Mensen vergelijken een aantal huizen en niet alle huizen in Nederland. Ook kiezen ze op basis van hoeveel spijt ze zouden krijgen door het niet te doen. Als laatste blijken mensen een huis dat in alle aspecten gemiddeld is fijner te vinden dan een huis dat iets heel goeds en iets heel slechts heeft. Als een huis bijvoorbeeld een mooie tuin heeft, maar een energielabel G, dan kiezen ze in de praktijk liever voor een gemiddelde tuin en een gemiddeld energielabel.

6.1.9 Enkel land en Rotterdam

Er is in dit onderzoek enkel gekeken naar de invloed van inbraakcijfers op de voorspellingen in Rotterdam en heel Nederland. Wellicht zijn er andere inzichten te vinden in andere steden en gebieden.

6.1.10 Niet alle postcodegebieden hebben evenveel inwoners

Er is nu uitgegaan van het aantal inbraken en pogingen op inbraak per postcodegebied, maar niet alle postcodegebieden hebben evenveel inwoners. Daarom is dit geen perfecte maatstaf voor het bepalen hoeveel criminaliteit er is rond een huis.

6.2 Mogelijke verbeteringen van het model

6.2.1 Reverse geocoding

Het is mogelijk om *reverse geocoding* te gebruiken om de postcode (en meer gegevens over de locatie, zoals de straatnaam en gemeente) te achterhalen op basis van de coördinaten. Hiervoor wordt data gebruikt van kaartbedrijven, zoals *OpenStreetMaps*. Er is geprobeerd om dit voor alle datapunten met een missende postcode te doen met behulp van de *library Nominatim* van *geopy.geocoders*. Echter duurt dit erg lang, omdat voor elk datapunt een aparte aanvraag gedaan moet worden richting de *server*. Nog erger is dat er een limiet is op het aantal aanvragen dat gedaan kan worden achter elkaar. Uit ervaring gebeurt dit soms al na tien aanvragen binnen een minuut. Om dit te voorkomen, zou gebruik gemaakt kunnen worden van *batch geocoding*. Dit is echter een betaalde dienst.

6.2.2 *Unsupervised clustering* van tevoren

Wellicht kan het helpen om *unsupervised clustering* methoden toe te passen en deze dan als *features* te gebruiken in de *supervised regression* modellen.

6.2.3 Meer modellen proberen

Niet alle mogelijke modellen zijn geprobeerd in dit onderzoek. Wellicht bestaan er andere modellen die nog geschikter zijn voor de taak van het schatten van huizenprijzen.

6.2.4 Trainen zonder uitschieters

Het kan een goed idee zijn om te trainen zonder de uitschieters. Deze beïnvloeden de resultaten namelijk drastisch. In de praktijk is het model waarschijnlijk toch niet bedoeld om uitschieters te voorspellen.

6.2.5 Zekerheid van voorspelling meegeven

Het kan een goed idee zijn om mee te geven hoe zeker het model is van de geschatte prijs. In onze implementatie van *nearest neighbours* kan bijvoorbeeld het aantal huizen dat binnen de radius lag, iets zeggen over hoe goed de voorspelling is. In het geval van uitschieters zit het model vaak fout. Als het een prijs voorspelt die erg anders is dan de gemiddelde prijs, kan het alvast meegeven dat er een goede kans is dat de voorspelling er ver naast zit.

6.2.6 Optimalisatie per stad

Het is nu al bekend dat de optimale parameters per stad heel anders zijn dan voor het hele land. Als nu voor elke stad alle voorspellingen worden gedaan op basis van de beste parameters voor die stad en daarna alle steden samengevoegd worden, zijn er betere resultaten over het hele land gezien.

6.2.7 *construction_type* meenemen

Er zit nog een erg interessante *feature* in de dataset die niet gebruikt is; namelijk *construction_type*. Hier staat of het huis een villa is, twee-onder-een-kap een rijtjeshuis en meer categorieën. Dit zegt een hoop over de prijs van het huis. Een mogelijkheid is om de categorieën met de hand een waarde te geven, maar het kan nog beter zijn om de gemiddelde prijs per huis voor elke categorie te berekenen en dat te gebruiken als extra *feature*. Probleem is wel dat bij sommige huizen meerdere categorieën genoemd worden. Dat maakt het lastiger.

6.2.8 Optimalisatie op basis van bias

Het is nu al mogelijk om een plot te maken met de bias per voorspelde prijs. Als hier met *polynomial regression* een kromme doorheen wordt getrokken, heb je een functie van de bias op basis van de voorspelde prijs. Als je dan deze uitkomst van deze functie van elke voorspelling aftrekt, zou de bias moeten verdwijnen en wordt hoogstwaarschijnlijk het model een stuk beter.

6.2.9 *Feature selection* algoritme

Nu is het kiezen van de *features* voor elk model met de hand gedaan, maar hier zijn ook algoritmes voor. Deze kunnen nog veel beter de invloed die modellen op elkaar hebben weergeven. Al is het maar het proberen van alle mogelijke combinaties. Dat zou het eindmodel al een stuk beter maken.

6.2.10 *Hyperparameter* selectie algoritme

Nu zijn de meeste *hyperparameters* gekozen door simpelweg een aantal mogelijkheden te proberen. Hier zijn ook algoritmen voor of er kunnen meer functies gemaakt worden, zoals wel bij de radius van *nearest neighbours* is gebeurd.

Literatuur

Cox, R., Brounen, D. & Neuteboom, P. (2015). Financial literacy, risk aversion and choice of mortgage type by households. *The Journal of Real Estate Finance and Economics*, *50*, 74–112.

Funda. (2023). *Huis kopen met behulp van kunstmatige intelligentie.* `https://www.funda.nl/meer-weten/kopen/een-huis-verkopen-met-behulp-van-kunstmatige-intelligentie/`. (Online; Geraadpleegd 29 Januari 2024)

KR&A. (2023). *About us.* `https://krafin.tech/about-kra/`. (Online; Geraadpleegd 29 Januari 2024)

Kreek, J. & Slechte, H. (2020). *Het oudste huis van nederland: 900 jaar proosdij in deventer.* Amsterdam University Press.

O'Sullivan, A. (2000). Urban economics. *Irwin Series in Economics*.

Politie. (2024). *Misdaad in kaart.* `https://www.politie.nl/mijn-buurt/misdaad-in-kaart?geoquery=Rotterdam&distance=25.0`. (Online; Geraadpleegd op 23 januari 2024)

Reitz, K. (2023). *Requests.* `https://pypi.org/project/requests/`.

Speller, C., Pol, P. & Mingardo, G. (2006). Veiligheid: beleving en beleid. *AGORA Magazine*, *22*(4).

Vallejo Velazquez, M., Kounadi, O. & Podor, A. (2020). Analysis and mapping of crime perception: A quantitative approach of sketch maps. *AGILE: GIScience Series*, *1*, 20.

voor de Statistiek, C. B. (2019). *Regionale kerncijfers nederland.*