Immobilienscout24 Analysis Notebook v2

1. Importing packages & the Immobilienscout24 scraped data

In [2]:
```
# !pip install pandas
# !pip install gensim
# !pip install spacy
# !pip install gensim
# !pip install matplotlib
# !pip install sklearn
# !pip install keras
# !pip install tensorflow
# !pip install scikit-learn
# !pip install scipy
# !pip install nltk
# !pip install random2
# !pip install warnings-plugin
# !pip install statsmodels
# !pip install seaborn
# !pip install DateTime
# !pip install datetime3

import gensim
import numpy as np
import spacy
from spacy import displacy
from gensim.corpora import Dictionary
from gensim.models import LdaModel
import matplotlib.pyplot as plt
import keras
import tensorflow
import statsmodels.api as sm
import sklearn
import sklearn.preprocessing
from sklearn.linear_model import LinearRegression
from scipy import stats
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF
import random
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)
from pandas import DataFrame, read_csv
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
```

```
import warnings
import os
warnings.filterwarnings('ignore')
%matplotlib inline

#For Scikit Regression:

from datetime import datetime
from sklearn import preprocessing
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
import seaborn as sb
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib notebook
```

```
unable to import 'smart_open.gcs', disabling that module
Using TensorFlow backend.
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/nicoweiner/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:

```
immodf = pd.read_csv("https://www.dropbox.com/s/04s1ptlkvy5l61s/FINAL_IMMOBILIENSCOUT24_03_05_20.csv?dl=1", sep=',', parse_dates=['Baujahr', 'Bezugsfrei_ab'], error_bad_lines=False, dtype='unicode', lineterminator='\n')
pd.options.display.max_columns = None
immodf.head(5)
```

Out[2]:

	ID	ScoutID	Typ	Kaufpreis	Provision	Energieeffizienzklasse	Objel
0	115756801	Objekt-Nr.: FB211 | Scout-ID: 115756801	Bungalow	518500.0	Nein	A+	
1	116853306	Objekt-Nr.: 562020 | Scout-ID: 116853306	Einfamilienhaus (freistehend)	301667.0	Nein	NaN	
2	116959865	Objekt-Nr.: W16091 | Scout-ID: 116959865	Einfamilienhaus (freistehend)	225900.0	Nein	A+	
3	116953174	Scout-ID: 116953174	Einfamilienhaus (freistehend)	218659.0	Nein	NaN	
4	116953218	Scout-ID: 116953218	Einfamilienhaus (freistehend)	287200.0	Nein	NaN	

2. Run NFM Model to extract common topics in listing descriptions

2.1. Removing all German stopwords in the listing descriptions as we only want to extract the relevant words

In [3]:
```
stopWords = set (stopwords.words('german'))

print(len(stopWords))
print(stopWords)
```

```
232
{'jene', 'machen', 'das', 'die', 'hatten', 'meinen', 'muss', 'eine
s', 'damit', 'und', 'war', 'indem', 'welchen', 'können', 'des', 'h
ab', 'als', 'solcher', 'bei', 'am', 'welches', 'würde', 'allem', '
eure', 'ihre', 'dessen', 'in', 'welchem', 'nicht', 'zwischen', 'bi
n', 'manches', 'eine', 'seines', 'mein', 'wenn', 'hin', 'einig', '
jenen', 'anderer', 'ihm', 'manche', 'seine', 'ihnen', 'weiter', 'd
en', 'alle', 'musste', 'wirst', 'ihres', 'mancher', 'hier', 'deine
', 'meinem', 'dasselbe', 'ihr', 'werde', 'einiges', 'du', 'solches
', 'euer', 'kein', 'dazu', 'der', 'haben', 'einmal', 'eures', 'sol
l', 'werden', 'man', 'zum', 'einigen', 'es', 'jeder', 'doch', 'and
ers', 'dass', 'jetzt', 'seinen', 'ihrem', 'was', 'ander', 'aber',
'derselbe', 'dann', 'jedes', 'unseres', 'meiner', 'wie', 'anderes'
, 'unsere', 'andere', 'deiner', 'einer', 'keine', 'keiner', 'sind'
, 'keinen', 'vor', 'anderem', 'bis', 'mich', 'sollte', 'ihrer', 'e
inem', 'nun', 'so', 'im', 'eurer', 'deinen', 'sich', 'noch', 'dies
en', 'denn', 'euren', 'seinem', 'unser', 'etwas', 'derer', 'unsere
n', 'anderm', 'anderr', 'ihren', 'sein', 'aus', 'gewesen', 'wieder
', 'wo', 'eurem', 'solche', 'aller', 'während', 'desselben', 'dies
er', 'zur', 'dieses', 'er', 'welcher', 'bist', 'viel', 'anderen',
'andern', 'diese', 'durch', 'meine', 'ich', 'deinem', 'wird', 'den
selben', 'ins', 'mit', 'von', 'sehr', 'gegen', 'für', 'würden', 'k
eines', 'manchen', 'auf', 'sonst', 'könnte', 'ihn', 'kann', 'sie',
'jener', 'wollen', 'dem', 'weil', 'zwar', 'nur', 'solchen', 'warst
', 'selbst', 'jedem', 'vom', 'ist', 'sondern', 'dir', 'demselben',
'welche', 'an', 'daß', 'einigem', 'seiner', 'diesem', 'zu', 'hatte
', 'ob', 'allen', 'unserem', 'einen', 'dort', 'mir', 'uns', 'manch
em', 'jenes', 'dieselben', 'oder', 'weg', 'da', 'euch', 'jeden', '
ohne', 'jede', 'solchem', 'unter', 'jenem', 'um', 'alles', 'dein',
'deines', 'einige', 'nach', 'wir', 'will', 'ein', 'über', 'dich',
'wollte', 'hinter', 'dies', 'nichts', 'habe', 'also', 'keinem', 'm
eines', 'waren', 'auch', 'hat', 'dieselbe', 'einiger', 'derselben'
}
```

In [4]:
```
tfidf_vect = TfidfVectorizer(max_df=0.8, min_df=2, stop_words=stopW
ords, token_pattern='[a-zA-Z][a-zA-Z]{2,}')
doc_term_matrix = tfidf_vect.fit_transform(immodf['Beschreibung'].v
alues.astype('U'))
```

2.2. Creating a probability matrix for topics

In [5]:
```
nmf = NMF(n_components=25, random_state=42)
nmf.fit(doc_term_matrix)
```

Out[5]:
```
NMF(alpha=0.0, beta_loss='frobenius', init=None, l1_ratio=0.0, max
_iter=200,
    n_components=25, random_state=42, shuffle=False, solver='cd',
tol=0.0001,
    verbose=0)
```

In [6]:
```
#retrieve the probability vector of words

first_topic = nmf.components_[1]
top_topic_words = first_topic.argsort()[-10:]
```

2.3. Extracting top words for 25 topics

In [7]:
```
#the words in the data grouped by topics

for i,topic in enumerate(nmf.components_):
    print(f'Top words for topic #{i}:')
    print([tfidf_vect.get_feature_names()[i] for i in topic.argsort
()[-15:]])
    print('\n')
```

```
Top words for topic #0:
['offene', 'giges', 'gig', 'dusche', 'kamin', 'wohnbereich', 'gart
en', 'ste', 'zugang', 'terrasse', 'wohn', 'gigen', 'essbereich', '
gige', 'gro']


Top words for topic #1:
['garageninnenhof', 'garagenraum', 'garagenkeller', 'garagenmiete'
, 'garagennebengeb', 'garagenneubau', 'garagennutzung', 'garagenpa
rkplatz', 'garagenparzelle', 'garagenpl', 'garagenplatz', 'garagen
platzes', 'garagenkomplex', 'zzt', 'nan']


Top words for topic #2:
['wfl', 'wunschimmobilie', 'notarkosten', 'preisg', 'montag', 'nac
hlass', 'ausgabe', 'immobilienversteigerungs', 'kataloginfos', 'ka
taloges', 'versteigert', 'eur', 'sichern', 'verkehrswert', 'telefo
nnummer']


Top words for topic #3:
['mieter', 'insgesamt', 'derzeit', 'eur', 'mieteinnahmen', 'garage
n', 'stellpl', 'drei', 'tze', 'jeweils', 'zwei', 'wohneinheiten',
'vermietet', 'mehrfamilienhaus', 'wohnungen']


Top words for topic #4:
['senden', 'interesse', 'eigent', 'pers', 'besichtigungstermin', '
nnen', 'vollst', 'immobilie', 'erhalten', 'expos', 'weitere', 'anf
rage', 'informationen', 'bitte', 'gerne']
```

```
Top words for topic #5:
['sanierungsbed', 'glichkeiten', 'besteht', 'anwesen', 'garagen',
'genutzt', 'garage', 'vorhanden', 'werkstatt', 'befindet', 'anbau'
, 'gro', 'scheune', 'wohnhaus', 'grundst']


Top words for topic #6:
['zzgl', 'keller', 'bietet', 'einliegerwohnung', 'einfamilienhaus'
, 'einbauk', 'cksfl', 'insgesamt', 'verteilt', 'grundst', 'betr',
'baujahr', 'nutzfl', 'wohnfl', 'che']


Top words for topic #7:
['heizung', 'modernisiert', 'eingebaut', 'komplett', 'jahre', 'san
iert', 'jahren', 'dach', 'jahr', 'neue', 'fenster', 'neu', 'erneue
rt', 'wurden', 'wurde']


Top words for topic #8:
['gepflegte', 'dhh', 'spitzboden', 'ruhiger', 'carport', 'grundst'
, 'moderne', 'stellplatz', 'neubau', 'garage', 'doppelhaus', 'haus
h', 'lften', 'lfte', 'doppelhaush']


Top words for topic #9:
['kostenlos', 'musterhaus', 'aussehen', 'wohnprojekt', 'besichtigt
', 'annonce', 'abgebildete', 'abgebildet', 'hauszeichnung', 'hausa
rten', 'traumhaus', 'hung', 'glich', 'mietkaufdirekt', 'mieterh']


Top words for topic #10:
['bauweise', 'rmepumpe', 'preis', 'nnen', 'nschen', 'bauen', 'enth
alten', 'schl', 'stein', 'nsche', 'grundst', 'ausstattung', 'kfw',
'inkl', 'user']


Top words for topic #11:
['wohnlage', 'nem', 'einliegerwohnung', 'befindet', 'wundersch', '
ner', 'terrasse', 'garage', 'nes', 'ruhiger', 'lage', 'einfamilien
haus', 'garten', 'nen', 'sch']


Top words for topic #12:
['lassen', 'ganze', 'wohnen', 'finden', 'genie', 'gen', 'ger', 'gl
ichkeiten', 'gend', 'ausreichend', 'gem', 'nnen', 'familie', 'biet
et', 'platz']


Top words for topic #13:
['balkon', 'zugang', 'terrasse', 'ste', 'wanne', 'abstellraum', 'd
iele', 'dusche', 'esszimmer', 'kinderzimmer', 'che', 'bad', 'wohnz
immer', 'schlafzimmer', 'flur']


Top words for topic #14:
```

```
['erreicht', 'hauses', 'dar', 'erstreckt', 'ebenfalls', 'ausgestat
tet', 'betreten', 'befindet', 'gen', 'treppe', 'gelangt', 'erreich
en', 'gelangen', 'verf', 'ber']


Top words for topic #15:
['besichtigung', 'testens', 'beantragt', 'rund', 'accesskey', 'sow
ie', 'objektnr', 'app', 'portal', 'immoviewer', 'link', 'com', 'ru
ndgang', 'tour', 'https']


Top words for topic #16:
['familienhaus', 'frei', 'geh', 'maisonette', 'derzeit', 'kaltmiet
e', 'dachgeschoss', 'befindet', 'erdgeschoss', 'gro', 'obergeschos
s', 'balkon', 'zweifamilienhaus', 'vermietet', 'wohnung']


Top words for topic #17:
['liegt', 'steht', 'grundst', 'hlen', 'erdem', 'badezimmer', 'samt
', 'verkauf', 'zweist', 'beheizt', 'ckige', 'zimmern', 'preis', 'e
uro', 'immobilie']


Top words for topic #18:
['ste', 'befinden', 'wanne', 'keller', 'vorhanden', 'weiteres', 'e
rdgeschoss', 'obergeschoss', 'drei', 'dachgeschoss', 'weitere', 'b
alkon', 'che', 'bad', 'zimmer']


Top words for topic #19:
['ger', 'che', 'schlafzimmer', 'zugang', 'drei', 'weitere', 'dachg
eschoss', 'ume', 'badezimmer', 'befindet', 'obergeschoss', 'erdges
choss', 'befinden', 'zwei', 'sowie']


Top words for topic #20:
['nutzung', 'wurde', 'gerne', 'tzlich', 'weitere', 'zus', 'pkw', '
stellpl', 'verkauf', 'tze', 'insgesamt', 'steht', 'stehen', 'verf'
, 'gung']


Top words for topic #21:
['tzt', 'ume', 'zentrum', 'genutzt', 'gewerbeeinheiten', 'gewerbe'
, 'ladenlokal', 'nutzfl', 'lage', 'vermietet', 'gewerbefl', 'gewer
beeinheit', 'wohn', 'ftshaus', 'gesch']


Top words for topic #22:
['kwh', 'ehemalige', 'befinden', 'baujahr', 'denkmalschutz', 'chen
', 'genutzt', 'stallgeb', 'uden', 'hauptgeb', 'wohngeb', 'udes', '
nebengeb', 'geb', 'ude']


Top words for topic #23:
['voll', 'wohneinheiten', 'hierbei', 'energieausweis', 'erstellt',
'zweifamilienhaus', 'wurde', 'betr', 'zustand', 'befindet', 'geh',
'einfamilienhaus', 'angebotenen', 'handelt', 'objekt']
```

```
Top words for topic #24:
['zustand', 'gebaut', 'sofort', 'ren', 'befindet', 'direkt', 'gart
en', 'liegt', 'garage', 'voll', 'gibt', 'keller', 'geh', 'unterkel
lert', 'haus']
```

In [8]:
```
# Append new column with topics as numerical values
test = immodf

topic_values = nmf.transform(doc_term_matrix)
test['Topic'] = topic_values.argmax(axis=1)
test.head(3)
```

Out[8]:

	ID	ScoutID	Typ	Kaufpreis	Provision	Energieeffizienzklasse	Objel
0	115756801	Objekt-Nr.: FB211 | Scout-ID: 115756801	Bungalow	518500.0	Nein	A+	
1	116853306	Objekt-Nr.: 562020 | Scout-ID: 116853306	Einfamilienhaus (freistehend)	301667.0	Nein	NaN	
2	116959865	Objekt-Nr.: W16091 | Scout-ID: 116959865	Einfamilienhaus (freistehend)	225900.0	Nein	A+	

2.4. Give suitable topics a title

Considering the output of the 25 topics above, we now select the ones that make sense and are suitable and categorize them with a name. The category name tells what a house description is focusing on. The ones that are do not make sense are later being dropped.

In [9]:
```
# Insert the output in dataset as new variables

test.Topic = test.Topic.astype('str')

test.Topic = test.Topic.replace(['1'],'garage')
test.Topic = test.Topic.replace(['2'],'house_auction')
test.Topic = test.Topic.replace(['3', '16'],'multi_dwelling')
test.Topic = test.Topic.replace(['4'],'house_viewing')
test.Topic = test.Topic.replace(['7'],'renovated')
test.Topic = test.Topic.replace(['13'],'house_layout')
test.Topic = test.Topic.replace(['5', '22'],'courtyard_estate')
```

In [10]:
```
test = test.astype('str')
```

In [11]:
```
test
```

Out[11]:

	ID	ScoutID	Typ	Kaufpreis	Provision	Energieeffizienzklasse
0	115756801	Objekt-Nr.: FB211 | Scout-ID: 115756801	Bungalow	518500.0	Nein	A+
1	116853306	Objekt-Nr.: 562020 | Scout-ID: 116853306	Einfamilienhaus (freistehend)	301667.0	Nein	nan
2	116959865	Objekt-Nr.: W16091 | Scout-ID: 116959865	Einfamilienhaus (freistehend)	225900.0	Nein	A+
3	116953174	Scout-ID: 116953174	Einfamilienhaus (freistehend)	218659.0	Nein	nan
4	116953218	Scout-ID: 116953218	Einfamilienhaus (freistehend)	287200.0	Nein	nan
...	...	...	...	...	...	...
48877	37000048	Scout-ID: 37000048	Besondere Immobilie	239000.0	Nein	nan
48878	36788963	Scout-ID: 36788963	Doppelhaushälfte	188500.0	Nein	nan
48879	36713355	Objekt-Nr.: 1477028 | Scout-ID: 36713355	Einfamilienhaus (freistehend)	169000.0	Nein	nan
48880	36652956	Scout-ID: 36652956	Bauernhaus	2650000.0	Nein	nan
48881	35588847	Scout-ID: 35588847	Bauernhaus	109000.0	Nein	nan

48882 rows × 32 columns

3. Import results of LIWC analysis & merge with Immoscout data

Outside of this notebook, the program LIWC2015 has been purchased which allowed to run multi-level linguistic analysis on the listing descriptions. The analysis has been conducted within the program LIWC2015, resulting in a new dataset with new columns that indicate the level of various linguistic features and sentiment. For more details on the particular linguistic feature, see: https://liwc.wpengine.com/compare-dictionaries/ (https://liwc.wpengine.com/compare-dictionaries/)

In [12]:
```
liwc1 = pd.read_csv("https://www.dropbox.com/s/yse4stsaf42aeo9/LIWC2015%20Results%20%28FINAL_IMMOBILIENSCOUT24_03_05_20.csv%29.csv?dl=1", sep=',', error_bad_lines=False, dtype='unicode', lineterminator='\n')
pd.options.display.max_columns = None
```

In [13]:
```
liwc2 = liwc1
liwc2 = liwc2.drop(['B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'AA', 'AB', 'AC', 'AD', 'AE', 'filler\r'], axis=1)
liwc2 = liwc2.astype('str')
liwc2
```

Out[13]:

	A	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	ppron	i
0	115756801	260	92.13	46.92	1.00	32.37	15.29	32.31	9.23	0.77	0.77
1	116853306	162	93.82	45.10	1.00	25.77	13.50	35.19	4.94	0.62	0.62
2	116959865	94	93.26	50.00	1.00	45.30	11.75	28.72	7.45	1.06	1.06
3	116953174	148	94.42	50.00	1.00	51.03	13.45	37.16	6.76	0.68	0.68
4	116953218	86	93.26	50.00	1.00	47.26	4.53	43.02	5.81	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...
56632	37000048	70	93.26	50.00	1.00	1.91	35.00	38.57	2.86	0.00	0.00
56633	36788963	87	94.26	50.00	1.61	25.77	12.43	27.59	20.69	0.00	0.00
56634	36713355	61	93.26	37.15	1.71	25.77	7.62	39.34	14.75	1.64	1.64
56635	36652956	71	93.26	55.59	1.00	25.77	5.46	39.44	7.04	1.41	0.00
56636	35588847	58	94.72	50.00	1.00	25.77	19.33	22.41	10.34	0.00	0.00

56637 rows × 57 columns

In [14]:
```
liwc2_merged = pd.merge(test, liwc2, how="left", left_on="ID", right_on="A")
```

In [15]:
```
liwc2_merged = liwc2_merged.drop(['A', 'ScoutID', 'Provision', 'Bezugsfrei_ab', 'Title', 'Beschreibung', 'Ausstattung', 'Lagebeschreibung', 'Sonstiges', 'PLZ', 'Ortsteil', 'ScrapingDate1', 'ScrapingDate2', 'ScrapingDate3', 'ScrapingDate4'], axis=1)
```

In [16]:
```
liwc2_merged
```

Out[16]:

	ID	Typ	Kaufpreis	Energieeffizienzklasse	Objektzustand
0	115756801	Bungalow	518500.0	A+	Erstbezug
1	116853306	Einfamilienhaus (freistehend)	301667.0	nan	Erstbezug
2	116959865	Einfamilienhaus (freistehend)	225900.0	A+	Erstbezug
3	116953174	Einfamilienhaus (freistehend)	218659.0	nan	Erstbezug
4	116953218	Einfamilienhaus (freistehend)	287200.0	nan	Erstbezug
...	...	...	...	...	...
53221	37000048	Besondere Immobilie	239000.0	nan	Modernisiert
53222	36788963	Doppelhaushälfte	188500.0	nan	Erstbezug
53223	36713355	Einfamilienhaus (freistehend)	169000.0	nan	Neuwertig
53224	36652956	Bauernhaus	2650000.0	nan	Modernisiert
53225	35588847	Bauernhaus	109000.0	nan	Renovierungsbedürftig

53226 rows × 74 columns

In [17]:
```
liwc3_merged = liwc2_merged.set_index('ID')
```

4. Dummifying categorical variables

This is necessary for the correlation and regression analysis. For each value of a categorical variable like "Objektzustand" (condition of the house), a new column is created with boolean values 0 = False and 1 = True.

In [18]:
```
dummydf = liwc3_merged

dummydf1 = pd.concat([dummydf, pd.get_dummies(dummydf['Typ'])], axis=1)
dummydf2 = pd.concat([dummydf1, pd.get_dummies(dummydf['Energieeffizienzklasse'])], axis=1)
dummydf3 = pd.concat([dummydf2, pd.get_dummies(dummydf['Objektzustand'])], axis=1)
dummydf4 = pd.concat([dummydf3, pd.get_dummies(dummydf['Baujahr'])], axis=1)
dummydf5 = pd.concat([dummydf4, pd.get_dummies(dummydf['Stadt'])], axis=1)
dummydf6 = pd.concat([dummydf5, pd.get_dummies(dummydf['Topic'])], axis=1)

dummydf_clean = dummydf6
dummydf_clean.head(3)
```

Out[18]:

	Typ	Kaufpreis	Energieeffizienzklasse	Objektzustand	Anzahl_zimmer
ID					
115756801	Bungalow	518500.0	A+	Erstbezug	3.0
116853306	Einfamilienhaus (freistehend)	301667.0	nan	Erstbezug	4.0
116959865	Einfamilienhaus (freistehend)	225900.0	A+	Erstbezug	3.0

Now, all categorical columns are dropped as we have them dummified.

In [19]:
```
dummydf_clean2 = dummydf_clean.drop(['Typ', 'Energieeffizienzklasse', 'Objektzustand', 'Baujahr', 'Stadt', 'Topic', 'nan', '26434', '35216', '39619', '52379', '52428', '52531', '73733', '87600', '88400', '98593', '0', '10', '11', '12', '14', '15', '17', '18', '19', '20', '21', '23', '24', '6', '8', '9', '5410', '5992'], axis=1)
```

In [20]:
```
dummydf_clean2
```

Out[20]:

ID	Kaufpreis	Anzahl_zimmer	Wohnflaeche (in m²)	Grundstueck (in m²)	timetosell	Flaeche_km2
115756801	518500.0	3.0	160.0	1200.0	4	nan
116853306	301667.0	4.0	135.0	860.0	4	65.04
116959865	225900.0	3.0	76.0	612.0	12	48.8
116953174	218659.0	6.0	144.0	550.0	12	nan
116953218	287200.0	4.0	141.0	801.0	12	nan
...	...	...	...	...	...	...
37000048	239000.0	10.0	212.0	646.0	12	23.4
36788963	188500.0	6.0	164.0	339.0	12	16.53
36713355	169000.0	3.0	96.0	380.0	12	nan
36652956	2650000.0	18.0	300.0	6000.0	12	39.6
35588847	109000.0	10.0	300.0	2000.0	12	nan

53226 rows × 10055 columns

The dataframe now has to be converted into integer format.

The pd.to_numeric function has to be run before and after using dropna, as dropna only removes NaNs if they are integer but also changes the datatype to object again.

In [21]:
```
clean_df = dummydf_clean2.apply(pd.to_numeric, downcast='integer',
errors='coerce')
clean_df = clean_df.dropna()
clean_df = clean_df.apply(pd.to_numeric, downcast='integer', errors
='coerce')
clean_df
```

Out[21]:

ID	Kaufpreis	Anzahl_zimmer	Wohnflaeche (in m²)	Grundstueck (in m²)	timetosell	Flaeche_km2
117501115	370450.00	4.0	140.00	731.0	8	221.05
117039799	330144.85	4.0	119.00	780.0	8	69.58
117432002	495900.00	5.0	147.00	600.0	8	328.48
116836252	507800.00	5.0	134.00	369.0	4	755.09
116390654	290037.00	5.0	148.42	0.0	12	165.63
...	...	...	...	...	...	..
47421561	3100000.00	7.0	270.00	1450.0	12	310.70
47421561	3100000.00	7.0	270.00	1450.0	12	310.70
47421561	3100000.00	7.0	270.00	1450.0	12	310.70
47421561	3100000.00	7.0	270.00	1450.0	12	310.70
46646607	698000.00	10.0	250.00	545.0	12	21.68

7984 rows × 10055 columns

As a result, we have a dataframe containing all suitable columns for a correlation matrix and following multiple linear regression.

In [22]:
```
# clean_df.to_csv('LIWC_Immoscout_full_integer_dataset.csv')
```

5. Correlation Matrix

Due to the large size of the dataset, the pandas correlation function would get stuck. Therefore, the more efficient numpy is utilized to create the correlation matrix.

In [23]:
```
def corr_closure(df):
    d = clean_df.values
    sums = d.sum(0, keepdims=True)
    stds = d.std(0, keepdims=True)
    n = d.shape[0]

    def corr(k=0, l=10):
        d2 = d.T.dot(d[:, k:l])
        sums2 = sums.T.dot(sums[:, k:l])
        stds2 = stds.T.dot(stds[:, k:l])

        return pd.DataFrame((d2 - sums2 / n) / stds2 / n,
                            clean_df.columns, clean_df.columns[k:l]
)

    return corr
```

In [24]:
```
corr = corr_closure(clean_df)

corr_matrix = corr(0, 10055)
```

In [25]:
```
y = pd.DataFrame(corr_matrix)
y
```

Out[25]:

	Kaufpreis	Anzahl_zimmer	Wohnflaeche (in m²)	Grundstueck (in m²)	timetosell	Flaeche_I
Kaufpreis	1.000000	0.354147	0.467444	0.060030	0.000727	0.151
Anzahl_zimmer	0.354147	1.000000	0.763244	0.052932	-0.025746	-0.023
Wohnflaeche (in m²)	0.467444	0.763244	1.000000	0.078738	-0.023292	0.010
Grundstueck (in m²)	0.060030	0.052932	0.078738	1.000000	-0.010602	-0.010
timetosell	0.000727	-0.025746	-0.023292	-0.010602	1.000000	0.003
...	...	...	...	...	...	
house_auction	-0.017901	0.000144	0.005028	-0.002133	-0.034269	-0.015
house_layout	-0.055269	-0.050174	-0.054426	0.000051	0.060013	-0.043
house_viewing	0.038060	0.030242	0.020052	0.013139	0.002472	-0.005
multi_dwelling	0.035442	0.309946	0.235191	-0.011153	-0.016528	-0.072
renovated	-0.023180	-0.016075	-0.035830	0.017032	-0.011057	-0.002

10055 rows × 10055 columns

In [26]:
```
# Replace all NaN with 0
x = y
y = y.replace([np.inf, -np.inf], np.nan)
```

In [27]:
```
xy = y.fillna(0)
```

In [28]:
```
# Only include correlations between two variables that are higher t
han 0.5.

m = (xy.mask(np.eye(len(xy), dtype=bool)).abs() > 0.5).any()
m
```

Out[28]:
```
Kaufpreis                False
Anzahl_zimmer             True
Wohnflaeche (in m²)       True
Grundstueck (in m²)      False
timetosell               False
                           ...
house_auction             True
house_layout             False
house_viewing            False
multi_dwelling            True
renovated                False
Length: 10055, dtype: bool
```

In [29]:
```
raw = xy.loc[m, m]
raw
```

Out[29]:

	Anzahl_zimmer	Wohnflaeche (in m²)	Flaeche_km2	Einwohner	Men	W
Anzahl_zimmer	1.000000	0.763244	-0.023120	-0.023687	-0.023411	-0.02
Wohnflaeche (in m²)	0.763244	1.000000	0.010618	0.018925	0.019163	0.01
Flaeche_km2	-0.023120	0.010618	1.000000	0.941563	0.941833	0.94
Einwohner	-0.023687	0.018925	0.941563	1.000000	0.999986	0.99
Men	-0.023411	0.019163	0.941833	0.999986	1.000000	0.99
Women	-0.023952	0.018694	0.941278	0.999987	0.999947	1.00
Population_density	-0.022696	0.003063	0.605092	0.763651	0.761792	0.76
Average_income	-0.075630	-0.061240	-0.006494	0.109368	0.106595	0.11
Analytic	-0.010587	-0.015324	-0.054170	-0.056885	-0.057069	-0.05
Clout	0.012753	0.012660	-0.035317	-0.042928	-0.042992	-0.04
Tone	-0.001279	-0.003243	0.029417	0.042003	0.041779	0.04
Sixltr	0.041358	0.052992	-0.004367	0.004172	0.004064	0.00
ppron	-0.019178	-0.045175	-0.000901	0.006331	0.006351	0.00
i	-0.028065	-0.052317	-0.009150	-0.001200	-0.001186	-0.00
affect	-0.043691	-0.051253	-0.004171	-0.002339	-0.002471	-0.00
posemo	-0.025780	-0.027780	0.024809	0.034363	0.034171	0.03
negemo	-0.036109	-0.045202	-0.035367	-0.043688	-0.043670	-0.04

anger	0.013452	0.025002	-0.016758	-0.019919	-0.019986	-0.01
social	-0.020884	-0.030289	-0.004229	0.003010	0.003038	0.00
family	0.010513	0.000985	-0.015230	-0.011015	-0.010912	-0.01
friend	-0.019539	-0.018850	0.037178	0.049549	0.049550	0.04
female	-0.003678	-0.001209	0.002150	0.003360	0.003364	0.00
male	-0.046302	-0.045887	-0.012816	-0.011422	-0.011537	-0.01
cogproc	-0.005883	-0.010927	0.020030	0.035738	0.035567	0.03
discrep	-0.033602	-0.030388	0.012719	0.010454	0.010396	0.01
tentat	-0.019512	-0.008427	0.036866	0.053192	0.052940	0.05
differ	-0.021958	-0.023497	-0.003920	0.014138	0.014018	0.01
percept	-0.028869	-0.006478	0.035268	0.043020	0.043120	0.04
see	-0.012351	0.012231	0.046975	0.049212	0.049320	0.04
hear	-0.015615	-0.012383	0.003374	0.011705	0.011729	0.01
drives	-0.005079	-0.024612	-0.017921	-0.012739	-0.012685	-0.01
affiliation	0.013187	0.001294	-0.009331	-0.000239	-0.000148	-0.00
achieve	-0.021500	-0.020419	-0.006520	-0.001761	-0.001650	-0.00
power	0.070434	0.063185	-0.012951	-0.003907	-0.003821	-0.00
risk	-0.051553	-0.067579	-0.028593	-0.035922	-0.035884	-0.03
focuspresent	0.017435	-0.016824	0.003074	-0.002108	-0.002239	-0.00
focusfuture	-0.016077	-0.009269	0.016366	0.015728	0.015745	0.01
leisure	-0.087808	-0.069427	-0.037923	-0.026189	-0.026028	-0.02
home	-0.071749	-0.062624	-0.012535	-0.009666	-0.009730	-0.00
relig	-0.014051	-0.001113	-0.009574	-0.012200	-0.012279	-0.01
death	0.049455	0.022263	0.007957	-0.004728	-0.004923	-0.00
informal	0.050700	0.040458	0.023977	0.028163	0.028140	0.02
swear	-0.021205	-0.009780	-0.000544	-0.004285	-0.004416	-0.00
netspeak	0.034734	0.030842	0.015890	0.023939	0.023885	0.02
nonflu	0.043785	0.029278	0.019727	0.017371	0.017436	0.01
Mehrfamilienhaus	0.456395	0.370329	-0.074441	-0.068350	-0.068037	-0.06
Wohnimmobilie (sonstige)	0.026814	0.018049	-0.026324	-0.020758	-0.020892	-0.02
1340	-0.003682	-0.000792	-0.009565	-0.007612	-0.007603	-0.00
Berlin	0.003472	0.048832	0.756583	0.846978	0.848784	0.84
Eichstätt	-0.005289	0.002844	-0.016569	-0.013185	-0.013171	-0.01
München	-0.052319	-0.038320	0.088910	0.268283	0.264920	0.27
garage	0.013745	0.026299	0.055812	0.056828	0.056977	0.05
house_auction	0.000144	0.005028	-0.015729	-0.010583	-0.010694	-0.01

multi_dwelling	0.309946	0.235191	-0.072327	-0.072448	-0.072165	-0.07

In [30]:
```
# Correlation matrix highlighting variables that have at least one strong correlation with another variable

correlation_matrix = raw.style.background_gradient(cmap='Blues', axis=0).set_precision(2)
correlation_matrix
```

Out[30]:

	Anzahl_zimmer	Wohnflaeche (in m²)	Flaeche_km2	Einwohner	Men	Wom
Anzahl_zimmer	1	0.76	-0.023	-0.024	-0.023	-0.(
Wohnflaeche (in m²)	0.76	1	0.011	0.019	0.019	0.(
Flaeche_km2	-0.023	0.011	1	0.94	0.94	0
Einwohner	-0.024	0.019	0.94	1	1	
Men	-0.023	0.019	0.94	1	1	
Women	-0.024	0.019	0.94	1	1	
Population_density	-0.023	0.0031	0.61	0.76	0.76	0
Average_income	-0.076	-0.061	-0.0065	0.11	0.11	0
Analytic	-0.011	-0.015	-0.054	-0.057	-0.057	-0.(
Clout	0.013	0.013	-0.035	-0.043	-0.043	-0.(
Tone	-0.0013	-0.0032	0.029	0.042	0.042	0.(
Sixltr	0.041	0.053	-0.0044	0.0042	0.0041	0.0(
ppron	-0.019	-0.045	-0.0009	0.0063	0.0064	0.0(
i	-0.028	-0.052	-0.0091	-0.0012	-0.0012	-0.0(
affect	-0.044	-0.051	-0.0042	-0.0023	-0.0025	-0.0(
posemo	-0.026	-0.028	0.025	0.034	0.034	0.(
negemo	-0.036	-0.045	-0.035	-0.044	-0.044	-0.(
anger	0.013	0.025	-0.017	-0.02	-0.02	-0
social	-0.021	-0.03	-0.0042	0.003	0.003	0.(
family	0.011	0.00098	-0.015	-0.011	-0.011	-0.(
friend	-0.02	-0.019	0.037	0.05	0.05	0
female	-0.0037	-0.0012	0.0022	0.0034	0.0034	0.0(
male	-0.046	-0.046	-0.013	-0.011	-0.012	-0.(
cogproc	-0.0059	-0.011	0.02	0.036	0.036	0.(
discrep	-0.034	-0.03	0.013	0.01	0.01	0.(
tentat	-0.02	-0.0084	0.037	0.053	0.053	0.(
differ	-0.022	-0.023	-0.0039	0.014	0.014	0.(

percept	-0.029	-0.0065	0.035	0.043	0.043	0.(
see	-0.012	0.012	0.047	0.049	0.049	0.(
hear	-0.016	-0.012	0.0034	0.012	0.012	0.(
drives	-0.0051	-0.025	-0.018	-0.013	-0.013	-0.(
affiliation	0.013	0.0013	-0.0093	-0.00024	-0.00015	-0.00(
achieve	-0.022	-0.02	-0.0065	-0.0018	-0.0016	-0.0(
power	0.07	0.063	-0.013	-0.0039	-0.0038	-0.(
risk	-0.052	-0.068	-0.029	-0.036	-0.036	-0.(
focuspresent	0.017	-0.017	0.0031	-0.0021	-0.0022	-0.(
focusfuture	-0.016	-0.0093	0.016	0.016	0.016	0.(
leisure	-0.088	-0.069	-0.038	-0.026	-0.026	-0.(
home	-0.072	-0.063	-0.013	-0.0097	-0.0097	-0.0(
relig	-0.014	-0.0011	-0.0096	-0.012	-0.012	-0.(
death	0.049	0.022	0.008	-0.0047	-0.0049	-0.0(
informal	0.051	0.04	0.024	0.028	0.028	0.(
swear	-0.021	-0.0098	-0.00054	-0.0043	-0.0044	-0.0(
netspeak	0.035	0.031	0.016	0.024	0.024	0.(
nonflu	0.044	0.029	0.02	0.017	0.017	0.(
Mehrfamilienhaus	0.46	0.37	-0.074	-0.068	-0.068	-0.(
Wohnimmobilie (sonstige)	0.027	0.018	-0.026	-0.021	-0.021	-0.(
1340	-0.0037	-0.00079	-0.0096	-0.0076	-0.0076	-0.0(
Berlin	0.0035	0.049	0.76	0.85	0.85	0
Eichstätt	-0.0053	0.0028	-0.017	-0.013	-0.013	-0.(
München	-0.052	-0.038	0.089	0.27	0.26	0
garage	0.014	0.026	0.056	0.057	0.057	0.(
house_auction	0.00014	0.005	-0.016	-0.011	-0.011	-0
multi_dwelling	0.31	0.24	-0.072	-0.072	-0.072	-0.(

In [31]:
```
# correlation_matrix.to_excel("Immo_correlation_matrix.xlsx")
```

In [32]:
```
# raw.to_csv('Immo_correlation_matrix.csv')
```

In [ ]:
```

```

6. SciKit Multiple Linear Regression (Price)

As we want to test how the top topics & LIWC linguistic features can be used to predict the price of a real estate listing, we drop all other columns before running the regression analysis.

In [33]:
```
RegressionDFpre = liwc2_merged

RegressionDF = pd.concat([RegressionDFpre, pd.get_dummies(RegressionDFpre['Topic'])], axis=1)

RegressionDF = RegressionDF.drop(['Typ', 'Energieeffizienzklasse', 'Objektzustand', 'Anzahl_zimmer', 'Wohnflaeche (in m²)', 'Grundstueck (in m²)', 'Baujahr', 'Stadt', 'Flaeche_km2', 'Einwohner', 'Men', 'Women', 'Population_density', 'Average_income', 'timetosell', 'Topic', '0', '10', '11', '12', '14', '15', '17', '18', '19', '20', '21', '23', '24', '6', '8', '9'], axis=1)

RegressionDF
```

Out[33]:

	ID	Kaufpreis	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	p
0	115756801	518500.0	260	92.13	46.92	1.00	32.37	15.29	32.31	9.23	
1	116853306	301667.0	162	93.82	45.10	1.00	25.77	13.50	35.19	4.94	
2	116959865	225900.0	94	93.26	50.00	1.00	45.30	11.75	28.72	7.45	
3	116953174	218659.0	148	94.42	50.00	1.00	51.03	13.45	37.16	6.76	
4	116953218	287200.0	86	93.26	50.00	1.00	47.26	4.53	43.02	5.81	
...	...	...	...	...	...	...	...	...	...	...	
53221	37000048	239000.0	70	93.26	50.00	1.00	1.91	35.00	38.57	2.86	
53222	36788963	188500.0	87	94.26	50.00	1.61	25.77	12.43	27.59	20.69	
53223	36713355	169000.0	61	93.26	37.15	1.71	25.77	7.62	39.34	14.75	
53224	36652956	2650000.0	71	93.26	55.59	1.00	25.77	5.46	39.44	7.04	
53225	35588847	109000.0	58	94.72	50.00	1.00	25.77	19.33	22.41	10.34	

53226 rows × 65 columns

In [35]:
```
RegressionDF = RegressionDF.apply(pd.to_numeric, downcast='integer'
, errors='coerce')
RegressionDF = RegressionDF.dropna()
RegressionDF = RegressionDF.apply(pd.to_numeric, downcast='integer'
, errors='coerce')
RegressionDF
```

Out[35]:

	ID	Kaufpreis	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	p
0	115756801	518500.0	260	92.13	46.92	1.00	32.37	15.29	32.31	9.23	
1	116853306	301667.0	162	93.82	45.10	1.00	25.77	13.50	35.19	4.94	
2	116959865	225900.0	94	93.26	50.00	1.00	45.30	11.75	28.72	7.45	
3	116953174	218659.0	148	94.42	50.00	1.00	51.03	13.45	37.16	6.76	
4	116953218	287200.0	86	93.26	50.00	1.00	47.26	4.53	43.02	5.81	
...	...	...	...	...	...	...	...	...	...	...	
53221	37000048	239000.0	70	93.26	50.00	1.00	1.91	35.00	38.57	2.86	
53222	36788963	188500.0	87	94.26	50.00	1.61	25.77	12.43	27.59	20.69	
53223	36713355	169000.0	61	93.26	37.15	1.71	25.77	7.62	39.34	14.75	
53224	36652956	2650000.0	71	93.26	55.59	1.00	25.77	5.46	39.44	7.04	
53225	35588847	109000.0	58	94.72	50.00	1.00	25.77	19.33	22.41	10.34	

52542 rows × 65 columns

In [36]:
```
#Delete outliers
z_price = np.abs(stats.zscore(RegressionDF))
threshold = 3
RegressionDF = RegressionDF [(z_price < 3).all(axis=1)]
```

In [37]:
```
RegressionDF = RegressionDF.set_index('ID')
```

In [38]:
```
RegressionDF = RegressionDF.apply(pd.to_numeric, downcast='integer'
, errors='coerce')
RegressionDF = RegressionDF.dropna()
RegressionDF = RegressionDF.apply(pd.to_numeric, downcast='integer'
, errors='coerce')
RegressionDF
```

Out[38]:

ID	Kaufpreis	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	ppron	
115756801	518500.00	260	92.13	46.92	1.0	32.37	15.29	32.31	9.23	0.77	0
116953218	287200.00	86	93.26	50.00	1.0	47.26	4.53	43.02	5.81	0.00	0
116917689	781900.00	192	95.00	47.92	1.0	34.83	16.00	32.81	13.02	0.52	0
117009990	680400.00	211	95.51	44.35	1.0	25.77	19.18	35.07	17.06	0.95	0
117039799	330144.85	125	93.98	46.80	1.0	40.12	17.86	34.40	8.80	0.80	0
...	...	...	...	...	...	...	...	...	...	...	
96704880	895000.00	92	95.80	45.67	1.0	45.75	11.50	38.04	13.04	1.09	1
96615346	439000.00	72	90.27	39.05	1.0	25.77	9.00	31.94	6.94	2.78	2
96486399	595000.00	323	88.65	41.43	1.0	31.03	10.42	41.49	9.29	2.17	2
96459385	865000.00	307	91.97	41.00	1.0	16.46	18.06	28.66	9.77	1.63	1
96457868	299000.00	126	79.28	31.71	1.0	14.75	9.69	26.98	8.73	4.76	4

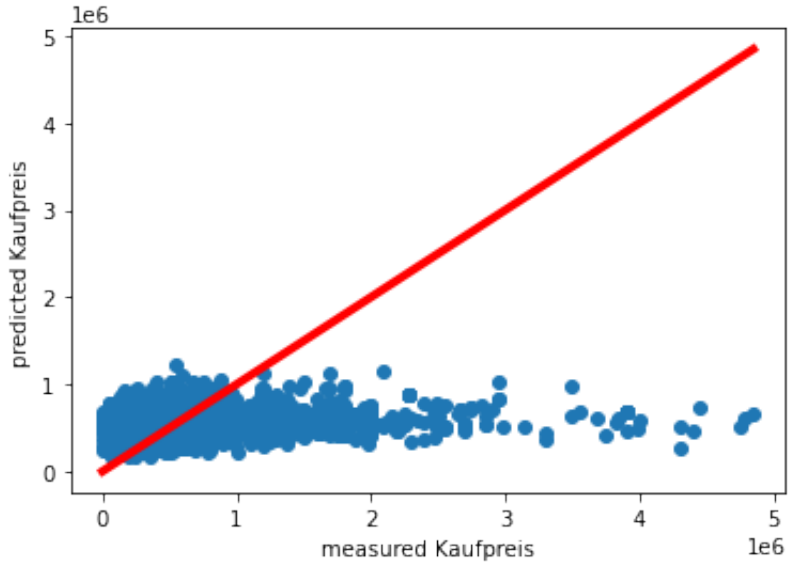
23028 rows × 64 columns

In [39]:
```
# First define dependent variable (Y) and all independent variables
(X) and create test and train set.

X = RegressionDF.drop('Kaufpreis', axis=1)
y = RegressionDF[['Kaufpreis']]


train_X, test_X, train_y, test_y = train_test_split(X, y, test_size
=0.2,
                                                    random_state=1)
```

In [40]:
```
# Now, select all independent variables and run a multiple linear r
egression.

multi_reg = LinearRegression()
multi_reg.fit(train_X, train_y)
y_predicted = multi_reg.predict(test_X)
print("Mean squared error: %.2f" % mean_squared_error(test_y, y_pre
dicted))
print ("Mean absolute error: %.2f" % mean_absolute_error(test_y, y_
predicted))
print('R²: %.2f' % r2_score(test_y, y_predicted))
```

```
Mean squared error: 233178827912.98
Mean absolute error: 303244.24
R²: 0.05
```

In [41]:
```
%matplotlib inline
fig, ax = plt.subplots()
ax.scatter(test_y, y_predicted)
ax.plot([test_y.min(), test_y.max()], [test_y.min(), test_y.max()],
'k-', color='red', lw=4)
ax.set_xlabel('measured Kaufpreis')
ax.set_ylabel('predicted Kaufpreis')
plt.show()
```



7. Tensorflow Neural Network (Price)

In [44]:
```
# Rename & extract the variable to predict
neuraldf = RegressionDF
target = neuraldf.Kaufpreis
neuraldf.drop(['Kaufpreis'],axis = 1 , inplace = True)
neuraldf['Target'] = target
```

In [45]:
```
# Split into train & test set whereby train set is 80% of the data.

n_train = round(neuraldf.shape[0] * 0.8)
train = neuraldf[:n_train]
test = neuraldf[n_train:]

target = train['Target']
train.drop(['Target'], axis = 1, inplace = True)

truth = test['Target']
test.drop(['Target'], axis = 1, inplace = True)
```

In [46]:
```
# Initiate a sequential model (i.e., no recurrence)
NN_model = Sequential()

# Make the input layer
NN_model.add(Dense(128, kernel_initializer='normal',input_dim = tra
in.shape[1], activation='relu'))

# Make hidden layers
NN_model.add(Dense(256, kernel_initializer='normal',activation='rel
u'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='rel
u'))

# Make the output layer
NN_model.add(Dense(1, kernel_initializer='normal',activation='linea
r'))

# Compile the network
NN_model.compile(loss='mean_squared_error', optimizer='adam', metri
cs=['mean_absolute_error'])
NN_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               8192
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 107,265
Trainable params: 107,265
Non-trainable params: 0
_________________________________________________________________
```

7.1. Checkpoint callback

A backup system that saves the models learned at each epoch. It saves only the models that are better than the previous models.

A file is made for each new model, containing the weights & biases of each neuron. Any of these files can be loaded to instantiate the corresponding network.

In [47]:
```
# Define how to name the files
checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
# Instantiate the checkpoint system
checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', v
erbose = 1, save_best_only = True, mode ='auto')
callbacks_list = [checkpoint]
```

7.2. Training the neural network

In [48]:
```
history = NN_model.fit(train, target, epochs=50,
             # No. of randomly sampled data points used to compute
the errors at each epoch (avoid overfitting)
             batch_size=32,
             # Size of validation set for cross-validation
             validation_split = 0.2,
             # Link to checkpoint system, to check the best model p
reviously built
             callbacks=callbacks_list)
```

```
Train on 14737 samples, validate on 3685 samples
Epoch 1/50
14737/14737 [==============================] - 1s 87us/step - loss
: 334173543029.9507 - mean_absolute_error: 363340.3750 - val_loss:
185991302922.3512 - val_mean_absolute_error: 282586.0938

Epoch 00001: val_loss improved from inf to 185991302922.35117, sav
ing model to Weights-001--185991302922.35117.hdf5
Epoch 2/50
14737/14737 [==============================] - 1s 81us/step - loss
: 275756730091.0329 - mean_absolute_error: 324074.4688 - val_loss:
186591927406.3197 - val_mean_absolute_error: 287132.2188

Epoch 00002: val_loss did not improve from 185991302922.35117
Epoch 3/50
14737/14737 [==============================] - 1s 81us/step - loss
: 275209827096.9625 - mean_absolute_error: 324119.5938 - val_loss:
181573202196.7718 - val_mean_absolute_error: 269544.0938

Epoch 00003: val_loss improved from 185991302922.35117 to 18157320
2196.77179, saving model to Weights-003--181573202196.77179.hdf5
Epoch 4/50
14737/14737 [==============================] - 1s 64us/step - loss
: 274326194176.9728 - mean_absolute_error: 323723.3750 - val_loss:
```

```
188770830062.0070 - val_mean_absolute_error: 294511.4688

Epoch 00004: val_loss did not improve from 181573202196.77179
Epoch 5/50
14737/14737 [==============================] - 1s 76us/step - loss
: 274141084295.1830 - mean_absolute_error: 323933.3438 - val_loss:
182801685605.4274 - val_mean_absolute_error: 275563.9688

Epoch 00005: val_loss did not improve from 181573202196.77179
Epoch 6/50
14737/14737 [==============================] - 1s 64us/step - loss
: 273543404622.4485 - mean_absolute_error: 324115.5625 - val_loss:
182759047928.2887 - val_mean_absolute_error: 275344.6875

Epoch 00006: val_loss did not improve from 181573202196.77179
Epoch 7/50
14737/14737 [==============================] - 1s 65us/step - loss
: 273184547032.2372 - mean_absolute_error: 322677.0625 - val_loss:
187753483388.7696 - val_mean_absolute_error: 291586.1875

Epoch 00007: val_loss did not improve from 181573202196.77179
Epoch 8/50
14737/14737 [==============================] - 1s 67us/step - loss
: 272920682170.8104 - mean_absolute_error: 323482.4688 - val_loss:
188152358266.7549 - val_mean_absolute_error: 292714.9062

Epoch 00008: val_loss did not improve from 181573202196.77179
Epoch 9/50
14737/14737 [==============================] - 1s 65us/step - loss
: 272841759318.8215 - mean_absolute_error: 322959.6562 - val_loss:
181489786615.4551 - val_mean_absolute_error: 269701.1875

Epoch 00009: val_loss improved from 181573202196.77179 to 18148978
6615.45508, saving model to Weights-009--181489786615.45508.hdf5
Epoch 10/50
14737/14737 [==============================] - 1s 79us/step - loss
: 271951558426.0047 - mean_absolute_error: 322024.6875 - val_loss:
186408615406.9102 - val_mean_absolute_error: 287916.6562

Epoch 00010: val_loss did not improve from 181489786615.45508
Epoch 11/50
14737/14737 [==============================] - 1s 87us/step - loss
: 271736158233.7094 - mean_absolute_error: 322013.4688 - val_loss:
192418442807.1599 - val_mean_absolute_error: 303352.2500

Epoch 00011: val_loss did not improve from 181489786615.45508
Epoch 12/50
14737/14737 [==============================] - 1s 77us/step - loss
: 271165566357.5838 - mean_absolute_error: 322652.4062 - val_loss:
184811222847.1490 - val_mean_absolute_error: 282260.4688

Epoch 00012: val_loss did not improve from 181489786615.45508
Epoch 13/50
14737/14737 [==============================] - 1s 65us/step - loss
: 270441239060.8108 - mean_absolute_error: 321041.4375 - val_loss:
189692895758.8668 - val_mean_absolute_error: 296077.1250
```

```
Epoch 00013: val_loss did not improve from 181489786615.45508
Epoch 14/50
14737/14737 [==============================] - 1s 68us/step - loss
: 270145854372.3493 - mean_absolute_error: 320627.9688 - val_loss:
192104844110.4326 - val_mean_absolute_error: 301889.2188

Epoch 00014: val_loss did not improve from 181489786615.45508
Epoch 15/50
14737/14737 [==============================] - 1s 65us/step - loss
: 269399358180.8488 - mean_absolute_error: 321258.1250 - val_loss:
184916361398.2914 - val_mean_absolute_error: 281460.3750

Epoch 00015: val_loss did not improve from 181489786615.45508
Epoch 16/50
14737/14737 [==============================] - 1s 64us/step - loss
: 269175840899.1876 - mean_absolute_error: 320260.0938 - val_loss:
189248515536.8988 - val_mean_absolute_error: 292733.0000

Epoch 00016: val_loss did not improve from 181489786615.45508
Epoch 17/50
14737/14737 [==============================] - 1s 64us/step - loss
: 268467699649.7415 - mean_absolute_error: 319956.7500 - val_loss:
190807612228.4288 - val_mean_absolute_error: 296433.7188

Epoch 00017: val_loss did not improve from 181489786615.45508
Epoch 18/50
14737/14737 [==============================] - 1s 65us/step - loss
: 268589392153.8310 - mean_absolute_error: 320237.3750 - val_loss:
187425864194.0841 - val_mean_absolute_error: 286622.1562

Epoch 00018: val_loss did not improve from 181489786615.45508
Epoch 19/50
14737/14737 [==============================] - 1s 66us/step - loss
: 268240918042.3000 - mean_absolute_error: 319576.9375 - val_loss:
187194747986.5313 - val_mean_absolute_error: 286198.0938

Epoch 00019: val_loss did not improve from 181489786615.45508
Epoch 20/50
14737/14737 [==============================] - 1s 67us/step - loss
: 267818985386.8114 - mean_absolute_error: 319905.5000 - val_loss:
194535451361.5023 - val_mean_absolute_error: 303963.3125

Epoch 00020: val_loss did not improve from 181489786615.45508
Epoch 21/50
14737/14737 [==============================] - 1s 68us/step - loss
: 267904420046.6483 - mean_absolute_error: 320027.2500 - val_loss:
196043123465.5175 - val_mean_absolute_error: 306422.4062

Epoch 00021: val_loss did not improve from 181489786615.45508
Epoch 22/50
14737/14737 [==============================] - 1s 65us/step - loss
: 267863771939.6632 - mean_absolute_error: 319674.8125 - val_loss:
185158863485.4643 - val_mean_absolute_error: 278376.1562

Epoch 00022: val_loss did not improve from 181489786615.45508
Epoch 23/50
14737/14737 [==============================] - 1s 65us/step - loss
```

```
: 267631693529.6617 - mean_absolute_error: 319915.2188 - val_loss:
185294966240.4602 - val_mean_absolute_error: 278641.0312

Epoch 00023: val_loss did not improve from 181489786615.45508
Epoch 24/50
14737/14737 [==============================] - 1s 64us/step - loss
: 267154401939.6903 - mean_absolute_error: 319049.0312 - val_loss:
183910544792.7664 - val_mean_absolute_error: 273875.1250

Epoch 00024: val_loss did not improve from 181489786615.45508
Epoch 25/50
14737/14737 [==============================] - 1s 63us/step - loss
: 266896495536.1618 - mean_absolute_error: 319162.5000 - val_loss:
188274022969.6608 - val_mean_absolute_error: 287212.9688

Epoch 00025: val_loss did not improve from 181489786615.45508
Epoch 26/50
14737/14737 [==============================] - 1s 66us/step - loss
: 266987118662.5967 - mean_absolute_error: 319079.2500 - val_loss:
196283438135.2988 - val_mean_absolute_error: 306622.2500

Epoch 00026: val_loss did not improve from 181489786615.45508
Epoch 27/50
14737/14737 [==============================] - 1s 65us/step - loss
: 267168164742.8877 - mean_absolute_error: 319882.2812 - val_loss:
196751185071.3444 - val_mean_absolute_error: 308302.4062

Epoch 00027: val_loss did not improve from 181489786615.45508
Epoch 28/50
14737/14737 [==============================] - 1s 75us/step - loss
: 266798864755.4666 - mean_absolute_error: 319172.5938 - val_loss:
190051302299.1284 - val_mean_absolute_error: 292035.2812

Epoch 00028: val_loss did not improve from 181489786615.45508
Epoch 29/50
14737/14737 [==============================] - 1s 68us/step - loss
: 266547752238.7460 - mean_absolute_error: 318726.7188 - val_loss:
199584580046.3978 - val_mean_absolute_error: 313420.1250

Epoch 00029: val_loss did not improve from 181489786615.45508
Epoch 30/50
14737/14737 [==============================] - 1s 72us/step - loss
: 266348292492.2033 - mean_absolute_error: 318736.6562 - val_loss:
200732374749.3340 - val_mean_absolute_error: 315442.4062

Epoch 00030: val_loss did not improve from 181489786615.45508
Epoch 31/50
14737/14737 [==============================] - 1s 64us/step - loss
: 266406783808.0130 - mean_absolute_error: 319310.5312 - val_loss:
186109301272.8705 - val_mean_absolute_error: 280641.0625

Epoch 00031: val_loss did not improve from 181489786615.45508
Epoch 32/50
14737/14737 [==============================] - 1s 62us/step - loss
: 266243552583.6216 - mean_absolute_error: 318126.0000 - val_loss:
182435293881.4871 - val_mean_absolute_error: 268509.3438
```

```
Epoch 00032: val_loss did not improve from 181489786615.45508
Epoch 33/50
14737/14737 [==============================] - 1s 61us/step - loss
: 266594570734.4550 - mean_absolute_error: 318909.4688 - val_loss:
185604926248.6404 - val_mean_absolute_error: 280983.7188

Epoch 00033: val_loss did not improve from 181489786615.45508
Epoch 34/50
14737/14737 [==============================] - 1s 61us/step - loss
: 265949120338.4266 - mean_absolute_error: 318070.1250 - val_loss:
191669721244.1704 - val_mean_absolute_error: 296672.8438

Epoch 00034: val_loss did not improve from 181489786615.45508
Epoch 35/50
14737/14737 [==============================] - 1s 68us/step - loss
: 266141765613.9339 - mean_absolute_error: 318522.8125 - val_loss:
184743189027.4301 - val_mean_absolute_error: 277642.0625

Epoch 00035: val_loss did not improve from 181489786615.45508
Epoch 36/50
14737/14737 [==============================] - 1s 75us/step - loss
: 265793737745.4407 - mean_absolute_error: 318247.2812 - val_loss:
188246751875.0220 - val_mean_absolute_error: 288302.8125

Epoch 00036: val_loss did not improve from 181489786615.45508
Epoch 37/50
14737/14737 [==============================] - 1s 62us/step - loss
: 265593906558.9317 - mean_absolute_error: 318080.7500 - val_loss:
194877075225.9126 - val_mean_absolute_error: 304482.3125

Epoch 00037: val_loss did not improve from 181489786615.45508
Epoch 38/50
14737/14737 [==============================] - 1s 60us/step - loss
: 265804181997.8992 - mean_absolute_error: 318197.2500 - val_loss:
199742692635.7189 - val_mean_absolute_error: 314440.3438

Epoch 00038: val_loss did not improve from 181489786615.45508
Epoch 39/50
14737/14737 [==============================] - 1s 60us/step - loss
: 265372785342.7710 - mean_absolute_error: 318675.5000 - val_loss:
184173095018.1514 - val_mean_absolute_error: 275232.8438

Epoch 00039: val_loss did not improve from 181489786615.45508
Epoch 40/50
14737/14737 [==============================] - 1s 62us/step - loss
: 265442548986.2154 - mean_absolute_error: 318162.8125 - val_loss:
187468342957.5381 - val_mean_absolute_error: 286681.3125

Epoch 00040: val_loss did not improve from 181489786615.45508
Epoch 41/50
14737/14737 [==============================] - 1s 62us/step - loss
: 265394425506.3516 - mean_absolute_error: 317869.5625 - val_loss:
189525603496.6752 - val_mean_absolute_error: 290974.9688

Epoch 00041: val_loss did not improve from 181489786615.45508
Epoch 42/50
14737/14737 [==============================] - 1s 63us/step - loss
```

```
: 265232205484.7744 - mean_absolute_error: 317737.2812 - val_loss:
191415152316.5438 - val_mean_absolute_error: 296483.4688

Epoch 00042: val_loss did not improve from 181489786615.45508
Epoch 43/50
14737/14737 [==============================] - 1s 61us/step - loss
: 264952978439.2959 - mean_absolute_error: 318471.4375 - val_loss:
186857437626.9460 - val_mean_absolute_error: 283226.4062

Epoch 00043: val_loss did not improve from 181489786615.45508
Epoch 44/50
14737/14737 [==============================] - 1s 61us/step - loss
: 265007359800.3002 - mean_absolute_error: 317752.7500 - val_loss:
189372350966.4130 - val_mean_absolute_error: 292362.6250

Epoch 00044: val_loss did not improve from 181489786615.45508
Epoch 45/50
14737/14737 [==============================] - 1s 59us/step - loss
: 265094638283.2783 - mean_absolute_error: 317748.7500 - val_loss:
190254549691.7102 - val_mean_absolute_error: 293538.0625

Epoch 00045: val_loss did not improve from 181489786615.45508
Epoch 46/50
14737/14737 [==============================] - 1s 61us/step - loss
: 265008837061.0420 - mean_absolute_error: 317942.1875 - val_loss:
185095046182.6258 - val_mean_absolute_error: 279819.7188

Epoch 00046: val_loss did not improve from 181489786615.45508
Epoch 47/50
14737/14737 [==============================] - 1s 58us/step - loss
: 264639269320.7942 - mean_absolute_error: 318081.1875 - val_loss:
200077409548.1574 - val_mean_absolute_error: 312966.3125

Epoch 00047: val_loss did not improve from 181489786615.45508
Epoch 48/50
14737/14737 [==============================] - 1s 58us/step - loss
: 264371507325.9068 - mean_absolute_error: 317154.3750 - val_loss:
189581198725.8703 - val_mean_absolute_error: 293077.2500

Epoch 00048: val_loss did not improve from 181489786615.45508
Epoch 49/50
14737/14737 [==============================] - 1s 59us/step - loss
: 263989607298.1628 - mean_absolute_error: 317403.9062 - val_loss:
184486909058.8830 - val_mean_absolute_error: 279869.8125

Epoch 00049: val_loss did not improve from 181489786615.45508
Epoch 50/50
14737/14737 [==============================] - 1s 77us/step - loss
: 264249051934.1044 - mean_absolute_error: 317177.5312 - val_loss:
201058087273.2483 - val_mean_absolute_error: 315721.9062

Epoch 00050: val_loss did not improve from 181489786615.45508
```

In [49]:
```
# Retrieve the hitory of every epoch's training
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail()
```

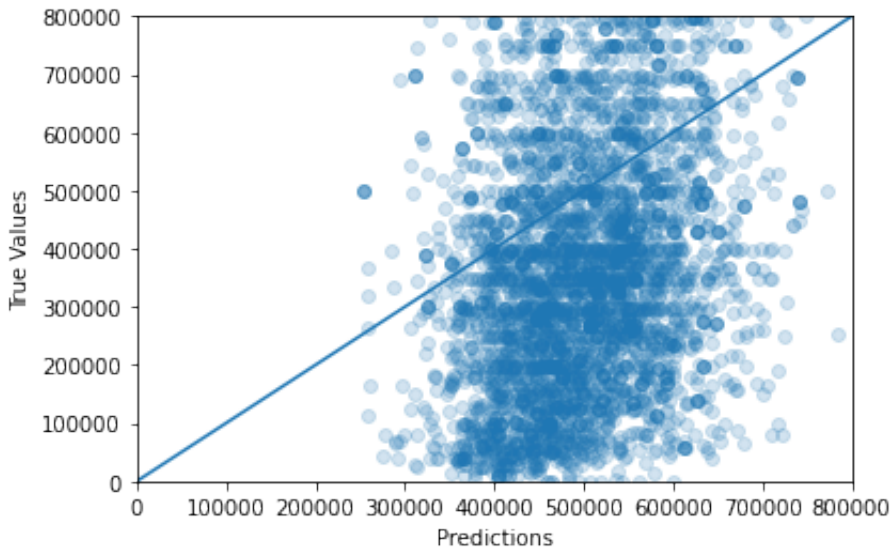
Out[49]:

	val_loss	val_mean_absolute_error	loss	mean_absolute_error	epoch
45	1.850950e+11	279819.71875	2.650088e+11	317942.18750	45
46	2.000774e+11	312966.31250	2.646393e+11	318081.18750	46
47	1.895812e+11	293077.25000	2.643715e+11	317154.37500	47
48	1.844869e+11	279869.81250	2.639896e+11	317403.90625	48
49	2.010581e+11	315721.90625	2.642491e+11	317177.53125	49

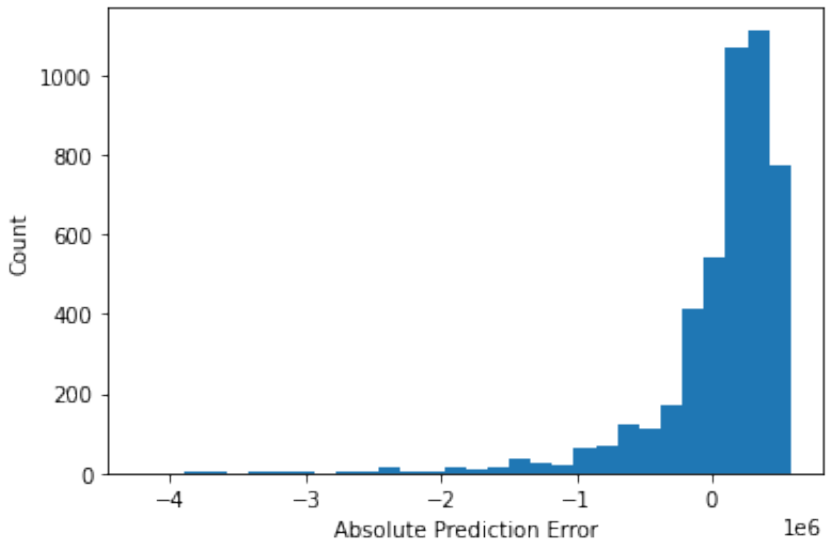
In [50]:
```
# Check best checkpoint in local folder (the last file saved)
#   Paste the name of the last file saved by the checkpoint system
weights_file = 'Weights-009--181489786615.45508.hdf5'

# Load the parameters of the models (weights & bias)
NN_model.load_weights(weights_file)
NN_model.compile(loss='mean_squared_error', optimizer='adam', metri
cs=['mean_squared_error'])
```

In [51]:
```
prediction = NN_model.predict(test)
```

In [52]:
```
%matplotlib inline
plt.figure()
plt.scatter(prediction, truth, alpha=0.2)
plt.xlabel('Predictions')
plt.ylabel('True Values')
lims = [0, 800000]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
plt.show()
```



In [53]:
```
%matplotlib inline
error = prediction - truth.to_numpy()
plt.figure()
plt.hist(error[0], bins = 30)
plt.xlabel("Absolute Prediction Error")
plt.ylabel("Count")
plt.show()
```



In [ ]: