Anja Kammer und Ido Sternberg

Vorhersage von kriminellen Vorfällen in San Francisco

Eine Projektarbeit im Rahmen der Lehrveranstaltung:
Aktuelle Kapitel sozialer Webtechnologien
- Maschinelles Lernen
von Prof. Dr. Christian Herta

Hochschule für Technik und Wirtschaft - Berlin

Inhaltsverzeichnis

1. Zusammenfassung

Mit Hilfe eines Klassifikators, sollen Arten von kriminellen Vorfällen – anhand von Zeit- und Ortsangaben ermittelt werden. Hierfür wird der von Kaggle.com vorgegebene Datensatz von unvollständigen Daten bereinigt und Attribute aufbereitet.
Anschließend erfolgt eine explorative Datenanalyse um mögliche Korrelationen zu erkennen und so die Auswahl geeigneter Features zu erleichtern.
Zur Veranschaulichung werden Diagramme offensichtliche Zusammenhänge verdeutlichen können.

Mit den 3 populärsten Klassifikatoren (Decision Tree, Random Forest, Naive Bayes) wird das Training durchgeführt. Nach der Validierung mittels des Kreuz- und Accuracy Algorithmus wird die Vorhersagegenauigkeit auf 22% geschätzt.

2. Der Auftrag / Zielstellung

Gegeben ist der Auftrag der Vorhersage von möglichem Auftreten einzelner Straftaten zu einer bestimmten Zeit an einem bestimmten Ort in San Francisco. Die Nutzung dieser Vorhersage könnte als Prävention von Straftaten genutzt werden.

Die Daten, die zur Erfüllung dieser Aufgabe nötig sind, stellt die Machine learning challenging Plattform Kaggle.com bereit. Hier ist es auch möglich, seine Ergebnisse mit anderen Teilnehmern zu vergleichen.

> Given time and location, you must predict the category of crime that occurred. We're also encouraging you to explore the dataset visually. (Kaggle.com, 2015)

> Gegeben sind Zeit und Ort, sage voraus welche Art der Straftat stattfindet. Wir ermutigen dich ebenso die Daten visuell zu entdecken. (Kaggle.com, 2015)

2.1. Der Datensatz

Dieser Datensatz enthält Vorfälle von Straftaten aus dem *SFPD Crime Incident Reporting System* für San Francisco. Die Datenerfassung reicht vom 01.01.2003 bis zum 13.05.2015.

Date	Zeitstempel der Straftat
Category	Art der Straftat. Diese Information soll vorhergesagt werden.
Descript	Beschreibung der individuellen Straftat
DayOfWeek	Wochentag des Auftretens der Straftat
PdDistrict	Bezirk der zuständigen Polizeiwache
Resolution	Beschreibung der Falllösung
Address	Die ungefähre Adresse der Straftat
X	Längengrad (geografische Koordinaten)
Y	Breitengrad (geografische Koordinaten)

2.2 Vorgehensweise

Code-Qualität
Während der gesamten Analyse werden Berechnungen und Vollständigkeit mit Hilfe von Assertions auf Plausibilität und geprüft. Des Weiteren wird Wert auf einen generischen Aufbau gelegt. Wann immer es möglich ist, werden Methoden angewendet um doppelten, und damit schlecht wartbaren, Code zu vermeiden.

Datenbereinigung
Zu Beginn wird die Datenmenge von unwichtigen Daten bereinigt, und einzelne Features werden, wenn notwendig, in Mehrere unterteilt.
(z.B.: Date => Monat, Jahr und Uhrzeit).

Explorative Datenanalyse
Aus diesem Teil erwächst die engere Featureauswahl. Durch die einfache Visualisierung der Daten, wird die Datenmenge verstanden und ein Gefühl für Korrelationen und Informationsgehalt entwickelt.

Featureauswahl
Durch eine Häufigkeitsanalyse und Ermittlung des Informationsgehaltes, werden aussagekräftige Features zum Training für den Klassifikator ausgewählt.

Algorithmen anwenden
Geeignete Algorithmen werden anhand der Datenlage ausgewählt und angewendet. Die Python Bibliothek *scikit-learn* stellt geeignete Klassifikatoren und Validierungsfunktionen bereit.

3. Datenvorverarbeitung

Vor Beginn der Explorativen Datenanalyse steht die Datenvorverarbeitung um eventuelle Fehler oder unvollständige Daten zu lokalisieren und zu bereinigen.

Da keine fehlenden Daten vorliegen, ist eine Schätzung unbekannter Werte oder das Setzen auf den Mittelwert, nicht notwendig. Die Aufgabe der folgenden Datenvorverarbeitung zielt eher auf die Bereinigung und Vorbereitung ab. Es werden Daten in Unterinformationen gesplittet, und offensichtlich irrelevante Daten ausgeschlossen.

Diese Datenvorverarbeitung wird mit *Scientific-Python* durchgeführt. Hierbei wird vor Allem die Bibliothek *Pandas* hilfreiche Funktionen zur Auswahl und Normierung der benötigten Features bereitstellen.

3.1 Bestimmung der Test- und Validierungsdaten

Vor jeder Analyse steht die Trennung der vorhandenen Datensammlung in Trainings- und Testdaten (inkl. Validierungsdaten) um Overfitting zu vermeiden.
Um eine eventuelle ungleichmäßige Verteilung der Werte zu verhindern, werden alle Datensätze zufällig durchmischt und anschließend verteilt. Diese Durchmischung erhalten wir durch das randomisieren der Indices aller Datensätze. Anschließend ist es unproblematisch die ersten 20% der Datensätze für das Testen und Validieren von der gesamten Datensammlung abzuschöpfen.

```
# shuffle the data
df = df.reindex(np.random.permutation(df.index))
# keep 80% of the data for training. The other 20% will be testing data
training_len = int(len(df)* 0.8)
testing_len = len(df) - training_len
df_train = df.head(training_len)
df_test = df.tail(testing_len)
```

3.2 Unvollständige Werte entfernen/skalieren

Jahr 2015

Die Datensammlung enthält Daten vom aktuellen Jahr 2015. Diese sind offensichtlich nicht vollständig. Eine Datenanalyse die einen jährlichen Anstieg/Fall der Kriminalität berechnet ist somit also nicht möglich, ohne dass es zu einer Verfälschung der Kriminalitätsrate kommt.

Zwei Lösungen kommen in Frage:

1. Skalierung der Daten, um eine Normalverteilung über die fehlenden Monate zu erhalten
2. Entfernung des laufenden Jahres aus der Datensammlung.

Im aktuellen Fall erscheint es sinnvoller die 2. Lösung zu bevorzugen, da eine Verfälschung eines möglichen Anstiegs/Abfall der Kriminalitätsrate auftreten kann.

```
# remove the data for the year 2015, which is incomplete
df = df[df.Year != 2015]
```

3.3 Datenaufbereitung

Date

Um eine genaue Analyse des Zeitpunkts eines Vorfalls durchzuführen, muss der Zeitstempel im Feld *Date* in Jahr, Monat und Uhrzeit aufgesplittet werden.
Eine mögliche Verwendung kann das Auswerten des Auftretens von Straftaten in einem bestimmten Zeitraum eines Tages, Monats oder Jahres sein.

Um später eine Auswertung der Straftatenverteilung aufs Jahr oder den Monat zu erhalten und eine tageszeitabhängige Relation aufzuzeigen, wurde das Attribut *Date* in eben jene Teile geparst.

Category

Unter allen Straftattypen, gibt es Ausreißer, die eine zu niedrige Datenmenge aufweisen. Das heißt: Diese Straftaten treten zu selten auf, um bei der Vorhersage signifikante Ergebnisse liefern zu können. Würde man mit diesen Daten die Kreuzvalidierung durchführen, erhielte man folgenden Fehler:

Warning: The least populated class in y has only 3 members, which is too few. The minimum number of labels for any class cannot be less than n_folds=5.
% (min_labels, self.n_folds)), Warning)

In diesem Schritt werden also die Straftaten die nicht häufiger als 100-mal auftreten im Zeitraum 2003-2014, entfernt. In diesem Fall ist das *trea* und *pornography/obscene mat.* Hierzu wurde ein sortiertes Histogramm der Auftrittshäufigkeit erstellt. (siehe Abbildung 7)

4. Explorative Datenanalyse

In diesem Teil wird durch die einfache Visualisierung der Daten, die Datenmenge verstanden und ein Gefühl für Korrelationen und Informationsgehalt entwickelt.
Aus der Explorativen Datenanalyse erwächst somit eine Auswahl aussagekräftiger Features zur Nutzung geeigneter Algorithmen.
Die folgenden Analysen basieren auf die Häufigkeitsverteilung von Features.
Das zu Grunde liegende Feature ist Jenes, Welches vorhergesagt werden soll: Die Straftatart (Category).

4.1 Visualisierungen

Straftaten je Bezirk

In welchen Bezirken finden welche Straftaten am Häufigsten/Seltensten statt. Am Anfang der Beantwortung dieser Frage, steht eine einfache Auswertung eines Histogramms in Balkendarstellung.

Aus diesem geht anscheinend hervor, dass Richmond – anders als Southern – der sicherste Bezirk ist. Allerdings muss auch hier die Relation beachtet werden. Denn die Anzahl der Vorfälle wird nicht mit der Einwohneranzahl in Verhältnis gesetzt.

Daher ist eine Ableitung von solchen Aussagen ohne die Kenntnis der Einwohneranzahl eher nicht zu empfehlen.

Verwendete Attribute: Category, PdDistrict

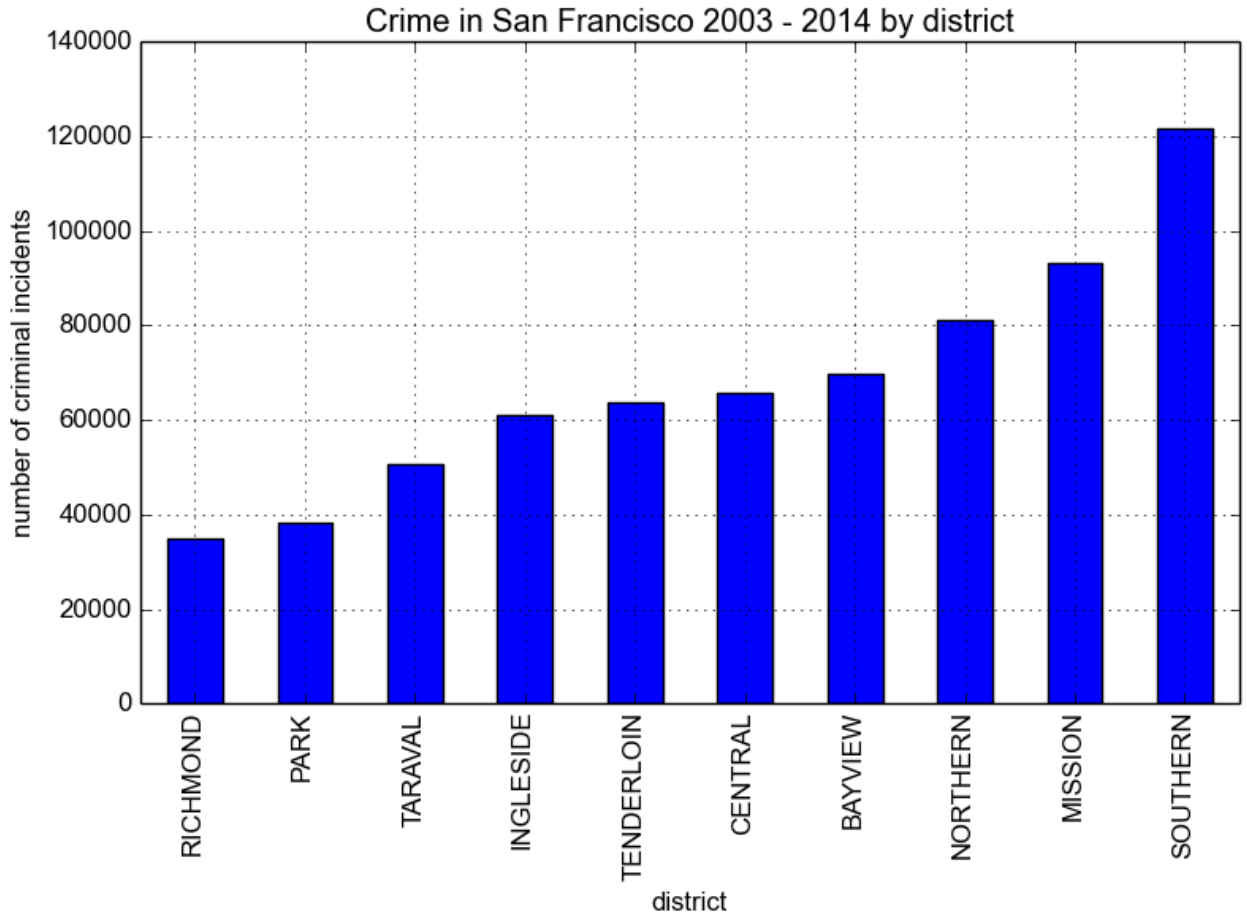

Abbildung 1 - Häufigkeitsverteilung aller Straftaten pro Bezirk

Entwicklungen im Verlauf der Jahre (Alle Vorfälle)
Gezeigt wird die Auftrittshäufigkeit als Liniendiagramm für jedes Jahr.
Hierbei ist zwar ein differenzierter Verlauf erkennbar, der aber durch die Skalierung des Diagramms verzerrt wurde.
Das bedeutet, das Auftreten blieb über die meiste Zeit stabil in einem engen Rahmen.
Verwendete Attribute: Category, Year

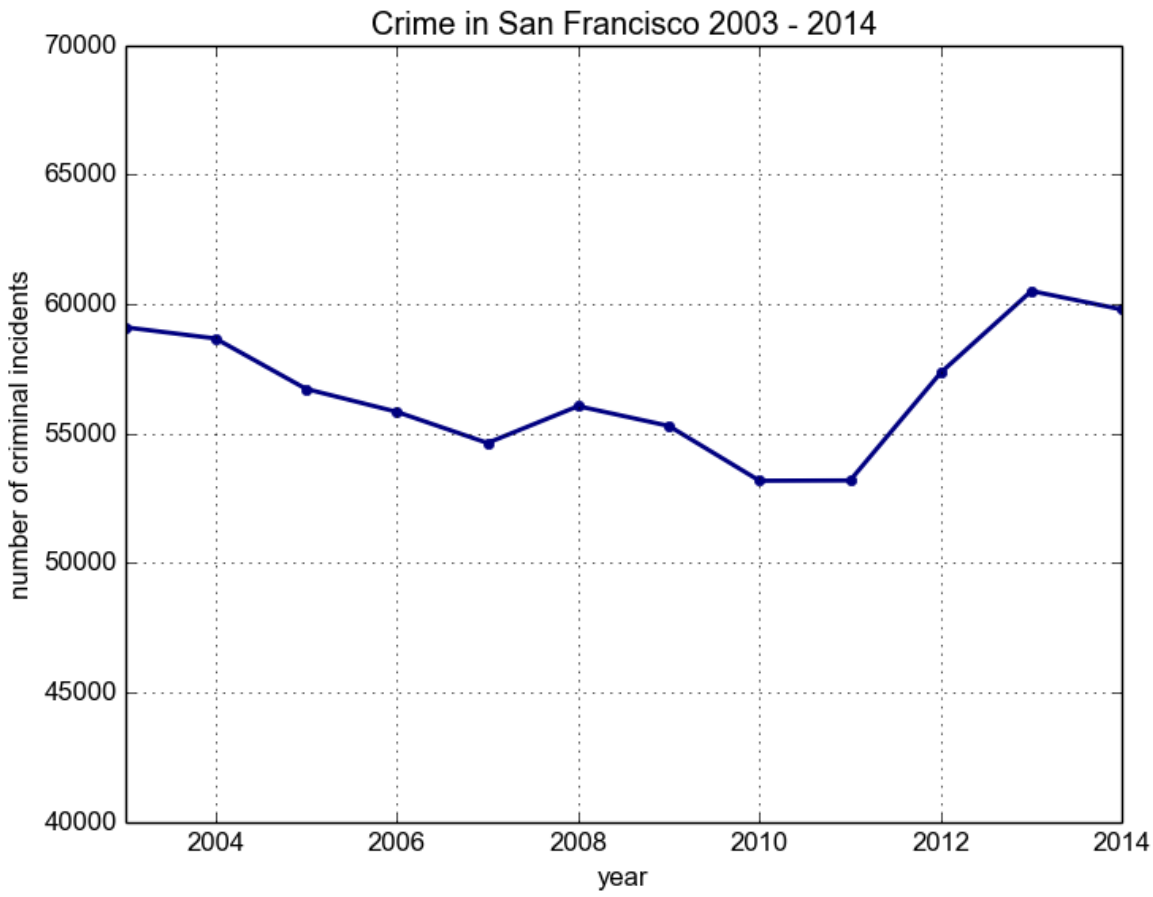

Abbildung 2 – 11 Jahresverlauf der Auftrittshäufigkeit krimineller Vorfälle

Entwicklungen im Verlauf der Jahre (Einzelne Vorfälle)
Die Darstellung der Trendentwicklung ist ebenso am Beispiel der Prostitution von kriminellen Aktivitäten in San Francisco interessant. Hier sehen wir einen stetigen Abwärtstrend über die Jahre ab 2007. Ob dies einen wirklichen Rückgang bedeutet oder einfach nur der Tolerierung seitens der Polizei geschuldet ist, kann nur vermutet werden.
Verwendete Attribute: Category, Year

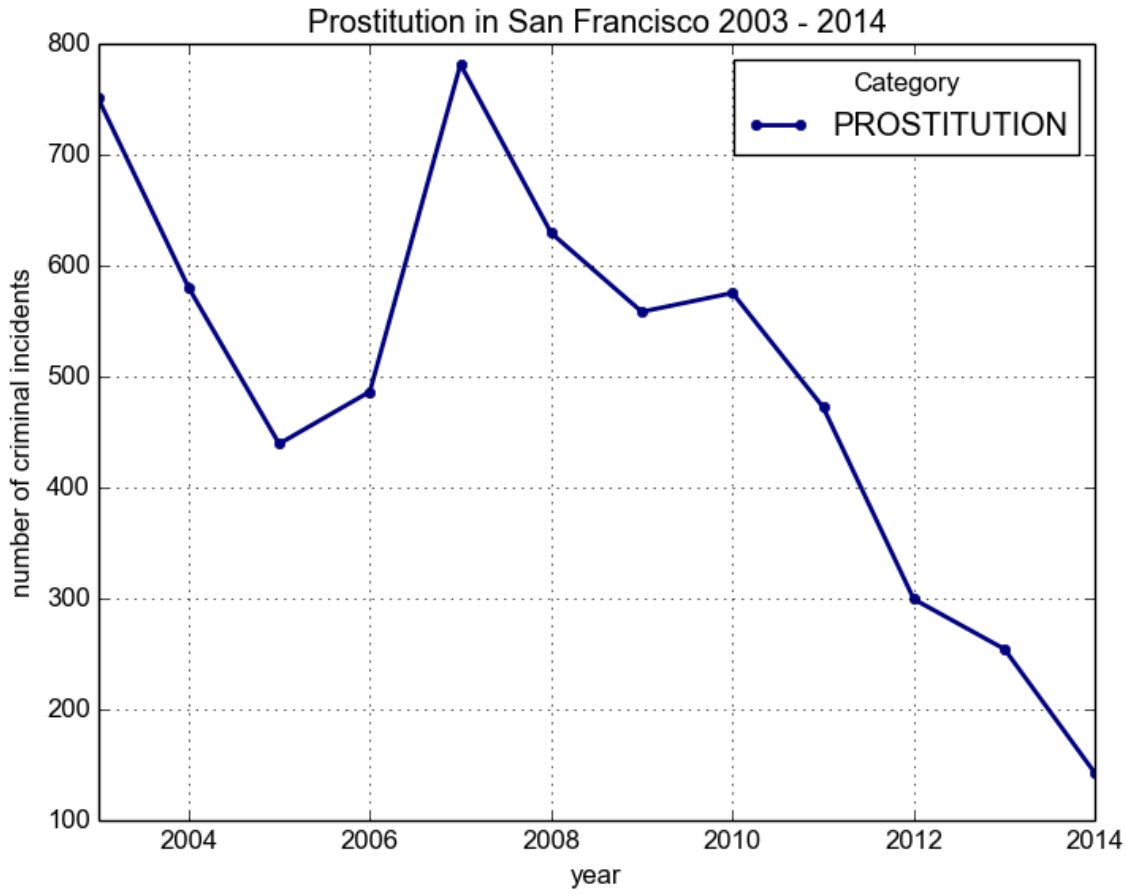

Abbildung 3 - Trendanalyse der Prostitutionsfälle im Verlauf der Jahre

Entwicklungen im Verlauf der Monate

Ebenso die Häufigkeitsverteilung im Balkendiagram aller Vorfälle auf die Monate bezogen, zeigen keine hohen Abweichungen untereinander. Allerdings ist hier zu beachten, dass Monatsdaten von 11 Jahre herangezogen werden, die ein eindeutiges Bild zeichnen. Es gibt 2 Peaks in der Entwicklung der Vorfälle im Jahr, das ist der Mai und Oktober.

Verwendete Attribute: Category, Month

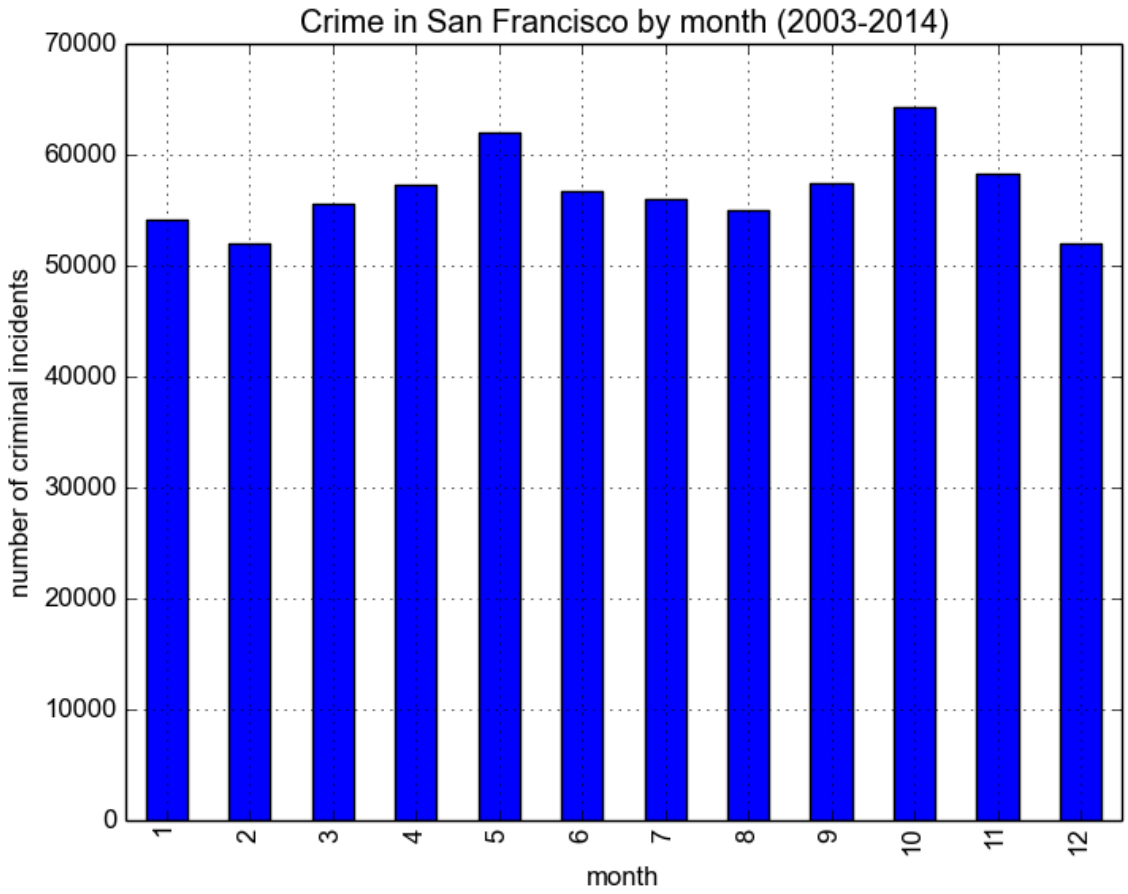

Abbildung 4 - Jahresverlauf der Auftrittshäufigkeit krimineller Vorfälle je Monat

Entwicklungen im Verlauf der Jahreszeiten
Das Balkendiagramm zur Auswertung der Auftrittshäufigkeit von kriminellen Vorfällen je Jahreszeit, bestätigt die Vorherige Aussage. Allerdings sind hier die Abtastraten zu gering um den deutlichen Anstieg und Fall der Kriminalitätsrate zum Mai und Oktober anzuzeigen. Die Daten erscheinen verwaschen und nicht scharf genug.
Verwendete Attribute: Category, Month

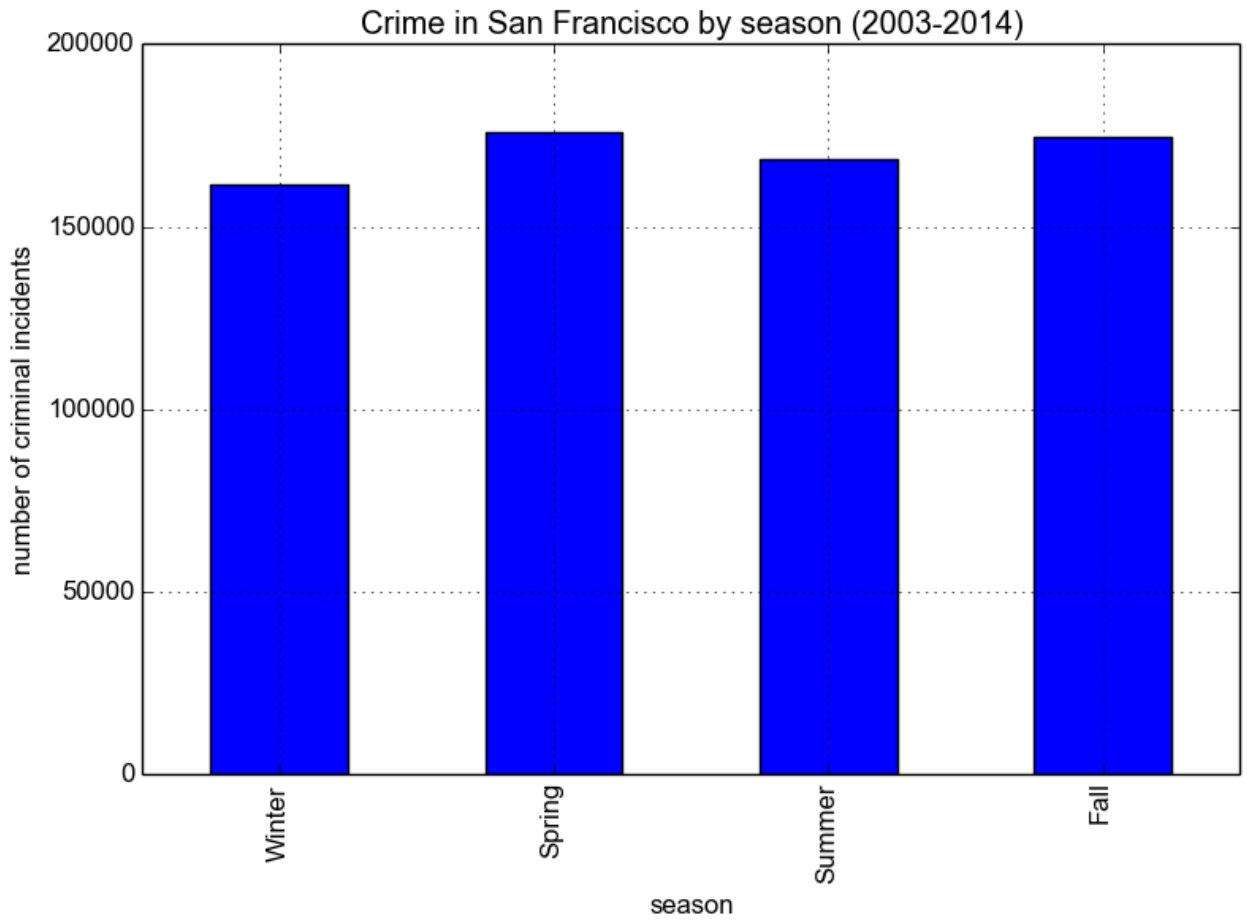

Abbildung 5 - Auftrittshäufigkeit von kriminellen Vorfällen je Jahreszeit

Entwicklungen im Verlauf der Jahreszeiten und Wochentage

Ein weiteres Balkendiagramm zeigt zusätzlich, zur Information des Auftretens je Jahreszeit, auch das Auftreten nach Wochentag. Eine offensichtliche Information: Freitags wird es krimineller als Sonntags. Um zu dieser Erkenntnis zu gelangen allerdings, braucht es aber keiner Auflistung je Jahreszahl.

Verwendete Attribute: Category, Month, DayOfWeek

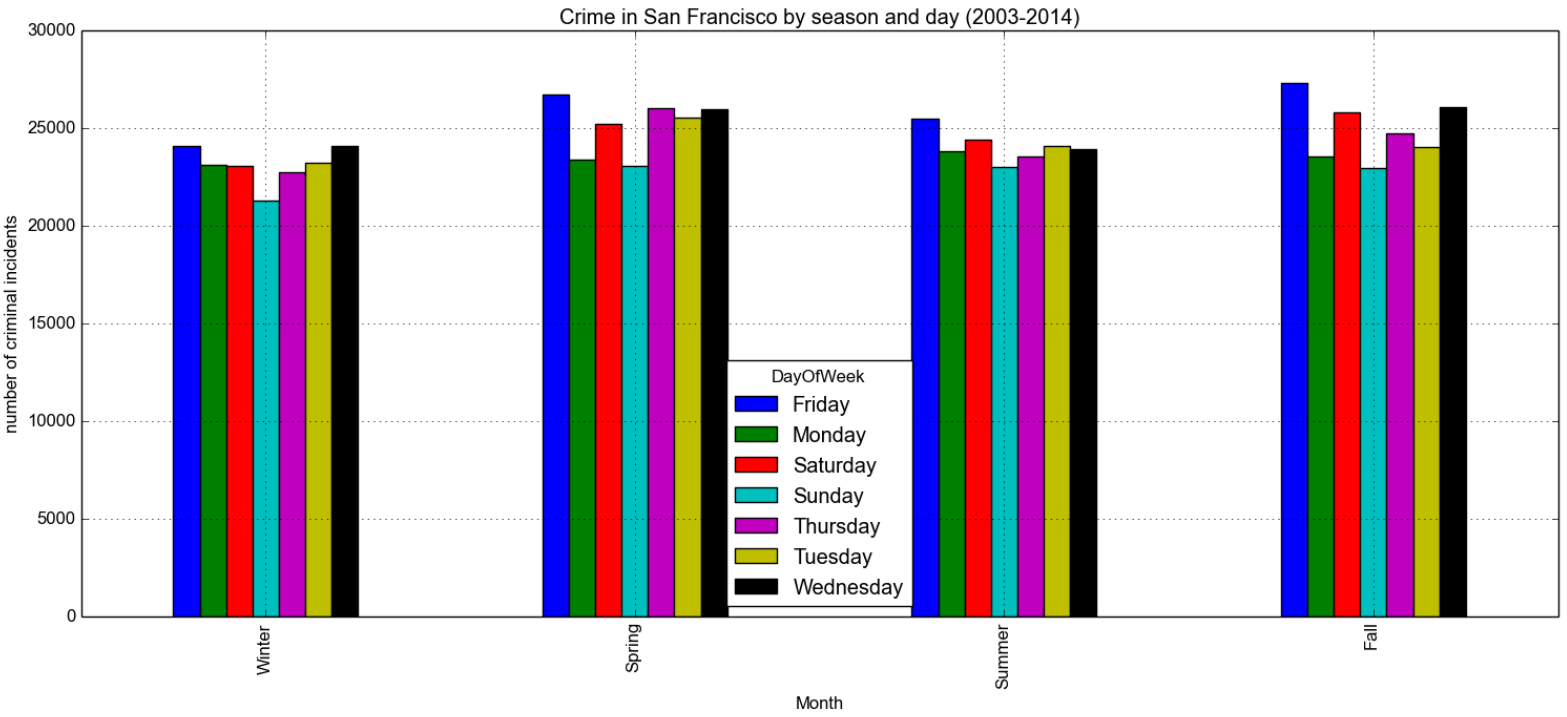

Abbildung 6 - Auftrittshäufigkeit von kriminellen Vorfällen je Jahreszeit und Wochentag

Auftrittsmengen-Histogramm

Das folgende Balkendiagramm zeigt die Top-Straftaten in San Francisco zusammen mit Jenen die weniger häufig vertreten sind.

In Kapitel 3.3 Datenaufbereitung haben wir bereits die beiden Schlusslichter trea und pornography/obscene mat. Als nicht-relevante Vorfälle identifiziert – die nicht in die Berechnung der Auftrittswahrscheinlichkeit einbezogen werden.

Verwendete Attribute: Category

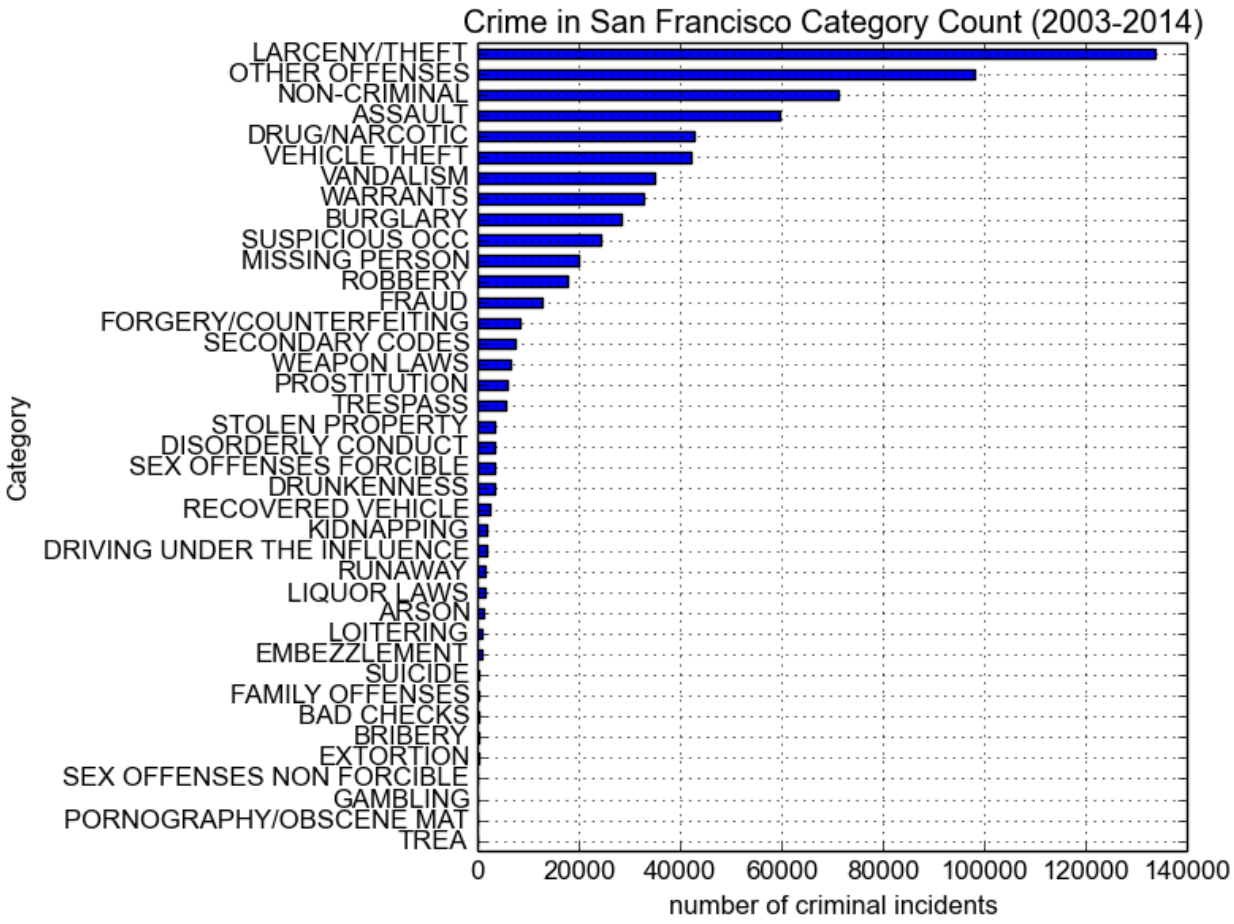

Abbildung 7 - Auftrittshäufigkeit je Straftatart

4.2 Bestimmung relevanter Features

Der Datenbestand kann redundante oder korrelierende Daten beinhalten, diese sind nicht aussagekräftig genug um in die Berechnungen eingeschlossen zu werden.

"Feature extraction and selection are the most important but underrated steps of machine learning. Better features are better than better algorithms." — Will Cukierski (Doing Data Science S. 176)

Es gibt verschiedene Modelle zur Auswahl geeigneter Features, neben der Entropie-Berechnung wie: P-Werte, AIC (Akaike Infomation Criterion), BIC (Bayesian Information Criterion).
In diesem Fall beschränken wir uns im Rahmen der Explorativen Datenanalyse auf die Häufigkeitsanalyse und der Entropie (Information-Gain), werfen aber auch einen Blick auf die Korrelation und deren Bedeutung in der Datenanalyse.

4.2.1 Häufigkeitsanalyse

Adresse

Die Art einer Straftat soll mit Hilfe des Tatorts vorhergesagt werden. Ist die genaue Adresse dafür geeignet? Um das herauszufinden haben wir ermittelt - wie eindeutig die Adressdaten sind. Von 850.500 Datensätzen sind 23143 Adressen eindeutig. Angaben wie der Bezirk sind hier einfacher zu unterscheiden, es gibt nur 10 Bezirke.

Das bedeutet: Das Trainieren des Klassifikators wird erfolgreicher sein, wenn dieser nur zwischen wenigen Fällen unterscheiden muss. Natürlich wird die Vorhersage dadurch auch nur eine gröbere Auflösung erreichen.

Um das zu kompensieren kann zur Visualisierung des Auftretens die Geo-Koordinaten (X, Y) verwendet werden, diese sind hochaufgelöst und ergeben daher eine hochauflösende Grafik für die Darstellung via Heatmap. Das ändert allerdings nichts an der eher groben Vorhersage.

Daher können die Adressdaten aus den Datensätzen entfernt werden.

Beschreibung

Die genaue Beschreibung der Straftat ist offensichtlich nicht für eine Klassifizierung geeignet, da diese zu detailliert ist um generalisiert zu werden. Aus diesem Grund wird die Beschreibung der Straftat aus dem Datensatz entfernt.

Wochentag

Der Wochentag ist nicht entscheidend bei dem Auftreten einer spezifischen Straftat.

Das bedeutet, täglich tritt dieselbe Art von Straftat in einer ähnlichen Häufigkeit auf. Mit einer Ausnahme: das Auftreten der Prostitutionsfälle unterscheidet sich stark, je nach Wochentag. Die Stärke des Rauschens ermitteln wir mit der Standart-Deviation. Hierfür holen wir uns eine Straftatart heraus und Listen das Auftreten je Tag an. Dieser Zeilenvektor der eine Abbildung eines Vorfalls im Datensatz darstellt, wird nun in einen Spaltenvektor Transponiert und die Pandas Funktion *describe()* ausgeführt. Wir erhalten dadurch eine statistische Auswertung der Daten. Hierbei fällt die Stärke des Rauschens zwischen den einzelnen Wochentagen auf. Dieser Wert könnte uns nun

dazu dienen, alle Straftaten aufzulisten, bei denen es bei der Häufigkeit des Auftretens einen Unterschied macht, an welchem Tag diese verübt wird. Aus diesem Grund bleibt der Wochentag Teil des auszuwertenden Datensatzes, um für diese Arten der Straftatbestände eine korrekte Vorhersage treffen zu können.

```
In [42]: display.head()
Out[42]:
DayOfWeek     Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  Sunday
Category
ARSON            228      235        200       199     220       220     211
ASSAULT        10560    10280      10553     10246   11160     11995   12082
BAD CHECKS        66       76         71        66      62        45      20
BRIBERY           41       37         40        39      49        42      41
BURGLARY        5262     5374       5457      5350    6327      4754    4231

In [43]: prostitution_row
Out[43]:
DayOfWeek      Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  Sunday
Category
PROSTITUTION      409     1421       1479      1547    1158       850     620

In [44]: prostitution_row.T.describe()
Out[44]:
       PROSTITUTION
count      7.000000
mean    1069.142857
std      449.735266
min      409.000000
25%      735.000000
50%     1158.000000
75%     1450.000000
max     1547.000000
```

Abbildung 8 - Auswertung von Prostitutionsfällen nach Wochentagen im Vergleich mit den ersten 5 Arten von Straftaten im Datensatz

4.2.2 Entropie / Information Gain

Die Entropie beschreibt wie vermischt die Daten sind. Ist eine 2er-Kombination von Features ähnlich, so ist die Entropie gering. ([0,1])

Je mehr Zufall im System ist, desto höher ist die Entropie. (Klinke, 2015)

Die Entropie wird zur Ermittlung des *Information-Gains* verwendet. Dieser gibt an, wie viel verwertbare Informationen in einer 2er-Kombination von Aussagen stecken.

```
def information_gain(p0, axis = 0):
  p = p0.copy()
  if axis == 1:
    p = p.T
  p_ = p.sum(axis=1)
  return entropy(p_) - conditional_entropy(p)
```

Abbildung 9 - verwendetes Code-Beispiel von:
http://christianherta.de/lehre/dataScience/machineLearning/decision-trees.php

Je überraschender das Ergebnis ist, desto höher ist der Informationsgehalt. (Herta, 2015)
Die Analyse des Informationsgehalts einiger Feature Kombinationen ergab folgendes sortierte Ergebnis:

Featurekombination	Information-Gain (absteigend)
Informationsgehalt(Category, PdDistrict)	0.095
Informationsgehalt(Category, Hour)	0.050
Informationsgehalt(Category, DayOfWeek)	0.005
Informationsgehalt(Category, Month)	0.002

Eine Aussage, die der Information-Gain nicht beantworten kann, ist wie viel Informationen in einer Feature-Kombination stecken. Da ein Training mit nur einem Features allerdings nicht üblich oder gar ausreichend ist, ist dieser Wert nur ein Anhaltspunkt zur Auswahl aller Features.

4.2.3 Auswertung

Mixen von Features um Ergebnisse am Ende zu vergleichen ist sinnvoll, und wird auch in Kapitel 5.3 Feature Kombinationen durchgeführt.
Dennoch ist die Vorauswahl nach oben genannten Gesichtspunkten auf folgende Features begrenzt worden:

PdDistrict, DayOfWeek, Month, Hour

4.3 Normierung

Nach der Auswahl geeigneter Features wird eine Normierung vorgenommen. Diese stellt eine Vektorisierung der einzelnen Werte dar. Mit Hilfe dieses Verfahrens wird eine Interpretation der Werte durch Nummerierung oder anderen Eigenschaften der Skalenniveaus ausgeschlossen. Diese Normierung erfolgt über die Funktion *get_dummies()* die von *Pandas* bereitgestellt wird.

```
def create_dummies(dataFrame, number_of_features):
  tmp_dist = pd.get_dummies(dataFrame['PdDistrict'])
  tmp_hour = pd.get_dummies(dataFrame['Hour'])
  tmp_day = pd.get_dummies(dataFrame['DayOfWeek'])
  tmp_month = pd.get_dummies(dataFrame['Month'])
  frames= [tmp_dist.T, tmp_hour.T, tmp_day.T, tmp_month.T]
  new_df = pd.concat(frames[0:number_of_features])
  new_df = new_df.T
  # Add The Category-number column
  new_df["Category"] = dataFrame["Cat_num"]
  return new_df

df_train_reduced = create_dummies(df_train_reduced, number_of_features)
df_test_reduced = create_dummies(df_test_reduced, number_of_features)
```

5. Vorhersage

Ziel dieses Projektes, ist das Vorhersagen der Möglichkeit Opfer einer bestimmten Straftat zu werden, in Bezug auf den Ort an dem man sich befindet.
Um solche Aussagen treffen zu können, wird ein Klassifikator verwendet. Dieser lernt mit Hilfe von gelabelten Trainingsdaten, mit welcher Wahrscheinlichkeit eine Klasse, aufgrund einer Datenlage, zugewiesen werden kann. Der Klassifikator kann also aufgrund eines Trainings, Muster erkennen und Klassen zuordnen.

5.1 Klassifikatoren

Verwendet wurden die 3 populärsten Klassifikatoren der scikit-learn Bibliothek:

Decision Tree

Ein Entscheidungsbaum ist eine geordnete Darstellung von Entscheidungsregeln, die graphisch an einen Baum erinnert. Der Algorithmus arbeitet Top-Down und hangelt sich entlang jeder Eigenschaft bis zur Wurzel.

Random Forest
Dieser Klassifikator besteht aus mehreren Entscheidungsbäumen, die nicht voreinander abhängen, also nicht korrelieren. Diese Bäume sind mittels Randomisierung während des Lernprozesses entstanden. So wird die Entscheidung von mehreren Bäumen getragen. Das Training mit dem Random Forest ist schneller als die Support Vector Machine, und daher effizient für große Datenmengen.
Ein weitere Vorteil darin liegt, dass der Random Forest – anders als der Decision Tree – Overfitting vermeiden kann.

Naive Bayes

Der Naive Bayes Klassifikator basiert auf dem Bayesschen Theorem und kann graphisch als Netz gesehen werden.
Er liefert nur gute Ergebnisse, solang die Features nicht zu stark korrelieren.

5.2 Validierung
Zur Überprüfen der Qualität des Trainings wurde die Validierung verwendet. Hiermit ist es möglich, die Ergebnisse verschiedenster Trainings-Modelle auf ihre Gültigkeit zu überprüfen

Kreuzvalidierung
Diese Validierungsform teilt die Menge aller Daten in diesem Fall in 5 Fragmente, und führt die Validierung 5-mal in einer anderen Kombination aus. Hieraus ergeben sich dann eben jene 5 Scores, je Fragment ein Score.

Accuracy Validierung
Auch mit Hilfe des *Accuracy classification scores* werden die Validierungsdaten mit der Vorhersage eines Algorithmus verglichen, und daraus eben jener Score gebildet.

Auswertung der Vorhersagegenauigkeit
Ein Vergleich aller Score-Werte der angewendeten Algorithmen ermittelt die Beste Wahl mit der Feature-Kombination: *PdDistrict, DayOfWeek, Hour*
Alle Trainingsverfahren erreichen eine 22%ige Vorhersagegenauigkeit.

Decision Tree
Cross-val-score:
[0.22244757, 0.22241656, 0.2235761, 0.22209565, 0.22300783]
Accuracy-score: 0.222362023247

Random Forest
Cross-val-score:
[0.22227856 0.22193889, 0.22316455, 0.22177227, 0.22261095]
Accuracy-score: 0.22147422141

Naive Bayes
Cross-val-score:
[0.22140411, 0.22288687, 0.22305431, 0.2220148, 0.22291229]
Accuracy-score: 0.221932821034

5.3 Feature Kombinationen

Testweise kann eine veränderte Kombination von Features durchgeführt werden. Hierbei ist allerdings zu beachten, dass aus Ermangelung an Features, lediglich eine Reduktion in der aufsteigenden Reihenfolge des Information-Gains sinnvoll erscheint. In diesem Fall ergab eine veränderte Kombination der Features folgende Ergebnisse:

2–Feature Training	**Accuracy Score**
Decision Tree:	0.223096958544
Random Forest:	0.222879417696
Naive Bayes:	0.221915182587

3–Feature Training	
Decision Tree :	0.223841635849
Random Forest:	0.223418328052
Naive Bayes:	0.222930348229

4–Feature Training	
Decision Tree :	0.207548944676
Random Forest:	0.204050796637
Naive Bayes:	0.222094185431

Anhand dieser Scores, wurde folgende finale Feature-Kombination ausgewählt:
PdDistrict, DayOfWeek, Hour
Entfernt wurde das Attribut *Month* welches einen Informations Gain von nur 0,002 besitzt.

```
'''
determine the number of features that will be used
for the prediction. sorted by Information-Gain (entropy.py)
'''
#number_of_features = 2   # features: 'PdDistrict', 'Hour'
number_of_features = 3    # features: 'PdDistrict', 'Hour', 'DayOfWeek'
#number_of_features = 4   # features: 'PdDistrict', 'Hour', 'DayOfWeek', 'Month'
if number_of_features > 4 or number_of_features < 2:
    raise ValueError("The number of features must be between 2 and 4")
```

6. Fazit

Die erzielte Vorhersagegenauigkeit von 22% erscheint zwar nicht befriedigend, könnte aber durch die Auswahl anderer Features gesteigert werden.

Ein kleiner Vergleich zeigt dennoch, dass selbst wenn bei der Vorhersage immer die Straftat die am Häufigsten auftritt (*Theft*) als Standart-Resultat vergeben würde, diese Vorhersage nur zu 19,7% korrekt wäre.
Daher verzeichnet das Training einen höheren Vorhersageerfolg, gegenüber der einfachen Schätzung des am Häufigsten auftretenden Vorfalls, um 2%.

Ebenfalls möglich wäre eine Aufnahme der GPS-Koordinaten in die Feature-Liste, die aber zu stark mit dem PdDistrict korrelieren würde, aufgrund dessen eben jenes Feature entfernt werden würde.
Im Allgemeinen waren die Resultate des Information-Gains sehr niedrig.
Max(Information-Gain) = 0,095

Auch bezüglich der Visualisierungen waren noch einige Fragen offen.
Wie sieht die Entwicklung der Kriminalität in jedem Bezirk aus, und welche Straftaten sind in jenen Bezirken hoch im Kurs?
Des Weiteren steht noch die Frage offen, welche Bezirke in San Francisco nun wirklich in Bezug auf die Einwohnerzahl besonders gefährlich/ungefährlich sind. Doch diese Daten müssten aus anderen Quellen, als die Gegebene, bezogen werden.

Verweise

(2015, 07 24). Retrieved from Kaggle: https://www.kaggle.com/c/sf-crime

Mckinney, W. (2012). *Python for Data Analysis.* Oreilly.

Milovanovic, I. (2013). *Python Data Visualization Cookbook.* Packt Publishing.

Schutt, R., & O'Neil, C. (2015). *Doing Data Science.* Oreilly.

Herta, C. (25. Juli 2015). Von ChristianHerta.de: http://christianherta.de/lehre/dataScience/entropy-informationGain.pdf abgerufen

Herta, C. (05. Juli 2015). *ChristianHerta.de*. Von http://christianherta.de/lehre/dataScience/machineLearning/decision-trees.php abgerufen

informatik.hu-berlin.de. (25. Juli 2015). Von http://www2.informatik.hu-berlin.de/~brueckne/ebl.pdf abgerufen

Klinke, S. (25. Juli 2015). Von WKlinke: http://filefarm.dyndns.org/wiki/index.php/Machine_Learning/Information_Gain abgerufen

Lohninger, H. (24. Juli 2015). *statistics4u.info*. Von http://www.statistics4u.info/fundstat_germ/cc_varsel_intro.html abgerufen

Lohninger, H. (24. Juli 2015). *statistics4u.info*. Von http://www.statistics4u.info/fundstat_germ/cc_cross_validation.html abgerufen

Lohninger, H. (24. Juli 2015). *statistics4u.info*. Von http://www.statistics4u.info/fundstat_germ/cc_corr_coeff.html abgerufen

Lohninger, H. (24. Juli 2015). *statistics4u.info*. Von http://www.statistics4u.info/fundstat_germ/cc_regression.html abgerufen

Loss, D. (24. Juli 2015). *Dirk-Loss.de.* Von http://www.dirk-loss.de/ipython-pandas-2013-05/datenanalyse-ipython-pandas.pdf abgerufen