Data Science Workshop Report

Aviv German, Ohad Cohen, Matan Akrabi

About the dataset:

For our Data Science project we chose to use dataset that contain tabular data of crimes that been occure in the city of Chicago.
The Chicago Crime dataset contains a summary of the reported crimes occurred in the City of Chicago from 2012 to 2017. Dataset been obtained from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system.
The dataset has 1,456,714 rows that every row represent diferent crime.
The dataset has been taken from Kaggle and can be found in the following link:
https://www.kaggle.com/datasets/currie32/crimes-in-chicago

The dataset contains the following columns:

- **ID:** Unique identifier for the record.
- **Case Number:** The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.
- **Date:** Date when the incident occurred.
- **Block:** address where the incident occurred.
- **IUCR:** The Illinois Uniform Crime Reporting code.
- **Primary Type:** The primary description of the IUCR code.
- **Description:** The frist report that been given along the IUCR code.
- **Location Description:** Description of the location where the incident occurred.
- **Arrest:** Indicates whether an arrest was made.
- **Domestic:** Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.
- **Beat:** Indicates the beat where the incident occurred. A beat is the smallest police geographic area – each beat has a dedicated police beat car.

- **District:** Indicates police district where the incident occurred.
- **Ward:** The ward (City Council district) where incident occurred.
- **Community Area:** Indicates the community area where the incident occurred. Chicago has 77 community areas.
- **FBI Code:** Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS).
- **X Coordinate:** The x coordinate of the location where the incident occurred in State of Illinois.
- **Y Coordinate:** The y coordinate of the location where the incident occurred in State of Illinois.
- **Year:** Year the incident occurred.
  Updated On: Date and time the record was last updated.
- **Latitude:** The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.
- **Longitude:** The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.
- **Location:** The location where the incident occurred.

Related work:

As we mention at the beginning of the report, we took the dataset from a Kaggle competition. We made an effort to choose a problem that has not yet been solved within the Kaggle competition and therefore we could not bring a notebook that deals with the same problem as our.

Instead, we brought three notebooks that we drew inspiration from, as for EDA (with the same data of course).

Crimes in Chicago By Facebooks Fbprophet Library - by PRAVIN

Chicago Crime Visualization by David

EDA of Crime in Chicago (2005 - 2016) - by Fhad

Define the data science prediction problem:

As we can see from the size of the dataset, the cops in the city of Chicago are very busy and handle a lot of crime cases. Chicago police wanted to maximize cases where arrest was made by prioritizing cases where there was a better chance they could make an arrest and ultimately optimize manpower better.
Therefore, we decided to investigate more than 1.4 million crime cases spread over 5 years, and to build for the Chicago Police a model that will help them deal with the above problem. Given a crime case, the model will know to predict whether or not an arrest will be made.

Naive ML model (serve as a baseline):

As our baseline model we chose to preform a basic preprocessing to our database that include data cleaning, handling imbalanced data, basic feature extraction and after the preeprocessing phase we trained basic naive model.

Data cleaning and balansing - We looked for the null values and deleted all the rows in the data that contain them. Which in hindsight was a mistake and was corrected at a later stage.

Another mistake we made was to balance the dataset by deleting samples. Of course, we later corrected this mistake as well.

Basic feature extraction - We took the columns of “Year”, “Primery Type” and “District” and apply One-hot-encoding on them and use them as a categorial features in our naive model.

Naive model - We trained a few algoritms and fully connected neural networks architectures and the best results were obtained from a fully connected neural network that been implemented from the Tensorflow.Keras library.

For the neural network architechture we used two hidden layers with the size of 6.
The activation function in the hidden layers was Rectifier Linear (ReLU) and for the the output neuron we used Sigmoid activation function.
For the loss funcion we used Binary Cross Entropy.
In addition, we used “Adam” optimizer (Stochastic Gradient Descent).

For hyper parameters: 10 Epochs, 32 Batch Size and 0.01 learning rate (Keras default).

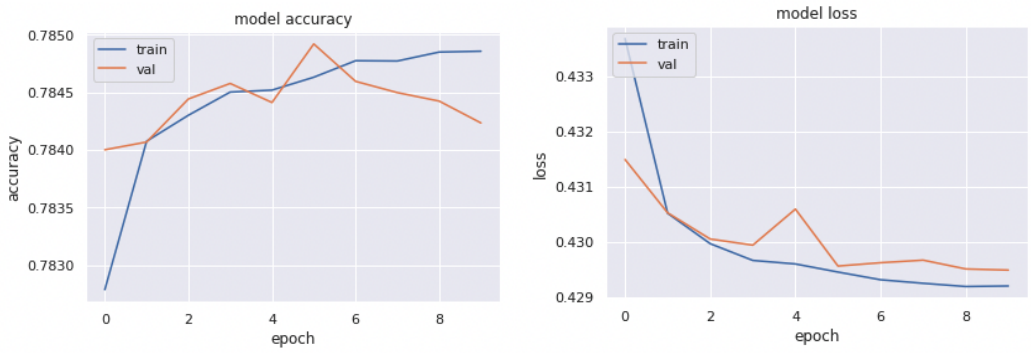
For validation set we use 10% from the training set.

Model Training Output:

```
# %%script echo skipping
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = tf.keras.models.Sequential()
classifier.add(tf.keras.layers.Dense(units=6, activation='relu'))
classifier.add(tf.keras.layers.Dense(units=6, activation='relu'))
classifier.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
history = classifier.fit(X_train, y_train, validation_split = 0.1, batch_size = 32, epochs = 10)
```

```
Epoch 1/10
91730/91730 [==============================] - 178s 2ms/step - loss: 0.4337 - accuracy: 0.7828 - val_loss: 0.4315 - val_accuracy: 0.7840
Epoch 2/10
91730/91730 [==============================] - 164s 2ms/step - loss: 0.4305 - accuracy: 0.7841 - val_loss: 0.4305 - val_accuracy: 0.7841
Epoch 3/10
91730/91730 [==============================] - 167s 2ms/step - loss: 0.4300 - accuracy: 0.7843 - val_loss: 0.4300 - val_accuracy: 0.7844
Epoch 4/10
91730/91730 [==============================] - 184s 2ms/step - loss: 0.4297 - accuracy: 0.7845 - val_loss: 0.4299 - val_accuracy: 0.7846
Epoch 5/10
91730/91730 [==============================] - 169s 2ms/step - loss: 0.4296 - accuracy: 0.7845 - val_loss: 0.4306 - val_accuracy: 0.7844
Epoch 6/10
91730/91730 [==============================] - 176s 2ms/step - loss: 0.4295 - accuracy: 0.7846 - val_loss: 0.4296 - val_accuracy: 0.7849
Epoch 7/10
91730/91730 [==============================] - 183s 2ms/step - loss: 0.4293 - accuracy: 0.7848 - val_loss: 0.4296 - val_accuracy: 0.7846
Epoch 8/10
91730/91730 [==============================] - 192s 2ms/step - loss: 0.4292 - accuracy: 0.7848 - val_loss: 0.4297 - val_accuracy: 0.7845
Epoch 9/10
91730/91730 [==============================] - 185s 2ms/step - loss: 0.4292 - accuracy: 0.7848 - val_loss: 0.4295 - val_accuracy: 0.7844
Epoch 10/10
91730/91730 [==============================] - 203s 2ms/step - loss: 0.4292 - accuracy: 0.7849 - val_loss: 0.4295 - val_accuracy: 0.7842
```

Model training process:



Model metrics for evaluation: 79% Accuracy.

```
Confusion Matrix:

[[380513  27335]
 [147529 259995]]
Accuracy: 79%
TPR: 72%
TNR: 90%
              precision    recall  f1-score   support

    Arrested       0.72      0.93      0.81    407848
Not Arrested       0.90      0.64      0.75    407524

    accuracy                           0.79    815372
   macro avg       0.81      0.79      0.78    815372
weighted avg       0.81      0.79      0.78    815372
```

Conclusions from the presentations in class:

- We searched for the null values and deleted all the rows in the data that contain them. Which in hindsight was a mistake and was corrected at a later stage (first presentation).

- It was wrong to balance the dataset by deleting samples. Of course, we later corrected this mistake as well (first presentation).

- We had to make use of the text of the report and extract features from it (first presentation).

- We had to make use of the date column and extract features from it (first presentation).

- If it takes us too much time to train the model due to the large number of examples, use less data (first presentation).

- To use "SHAP" to understand what are the strongest features that contribute to learning and what are the features that harm its performance (second presentation).

Appropriate ML model:

After Amit gave us his comments, we proceeded to implement them on our model.
We took the column of the date the crime was committed and created new features from it:

- Hour - A number between 0-23 that represent the hour during the day (we of course treated it as a categorical feature and not as a number).

- Part of the day - We took the "hour" feature and created a new feature from it that symbolizes the part of the day that the crime was committed (morning, noon, evening or night).

- Day - A number between 0-6 that represents the day of the week.

- Is Weekend - We took the "day" feature and created a new feature from it that would show whether the crime was committed in the middle of the week or at the weekend (of course we considered Saturday and Sunday as the weekend).

- Month - A number between 1-12 that represents the month.

After that, we took the text of the initial report, we cleaned it, arranged it and then extracted additional features from it:

- Character Count - Count the characters for each report.
- Words Count - Count the number of words in every report.
- Caracter Per Word - Character Count / Words Count.
- Special character - Count the special charaacter in the report (for example, if report contain the character '$').
- Numbers - Check if the report contain numbers.

At this stage, we felt that the text still had a lot of potential, so we decided to use it to train a model on its own (in addition to the first model). That's why we used the Bag-Of-Words approach to extract features from the text and finally train a model on it by itself.

* So from now on in the report we will call the model that deals only with the text "BOW model" and the model with all the other features "Main model".

We used One-hot-encodind on our categorial features and after that we scale the numeric features with standard Scaler.

Training appropriate model:

In this stage we had two datasets. One for the "main model" with 652 features (1456714, 652) , and the "BOW model" with 432 features (1456714, 432).
This is a large amount of parameters (thanks to One-hot-encoding) and it took us a lot of time to train the model.

To find the best hyper parameters for our fully connected neural netwoork we used Grid Search involved Cross Validation and we got the next results:

Hidden layers: 128 -> 64 -> 32 -> 16.
We use Dropout of 20% for regularization to prevent our networks from Overfitting. We used it between the two last hidden layers.

15 Epochs, 32 Batch Size and 0.01 learning rate (Keras default).
For the loss funcion we used Binary Cross Entropy and for the optimizer we used "Adam" (Stochastic Gradient Descent).

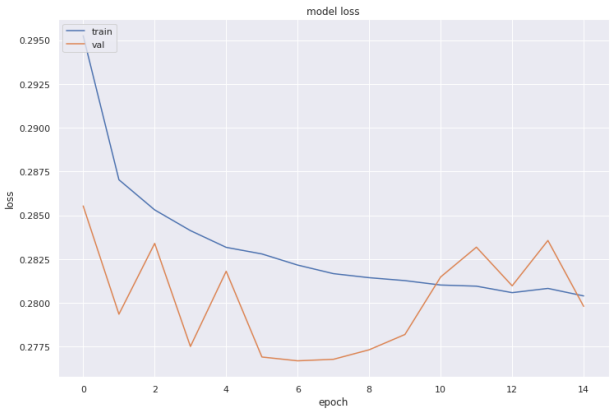
For validation set we use 10% from the training set.

For processing power we used a machine that contains 4 graphics cards of 2080Ti, significant GPU power without which we could not train the model.

Our neural network architecture:

```
def train_and_save_model(X_train, y_train, model_name):
    classifier = tf.keras.models.Sequential()
    classifier.add(tf.keras.layers.Dense(units=128, activation='relu'))
    classifier.add(tf.keras.layers.Dense(units=64, activation='relu'))
    classifier.add(tf.keras.layers.Dense(units=32, activation='relu'))
    classifier.add(tf.keras.layers.Dropout(0.2))
    classifier.add(tf.keras.layers.Dense(units=16, activation='relu'))
    classifier.add(tf.keras.layers.Dropout(0.2))
    classifier.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

    history = classifier.fit(X_train, y_train, validation_split = 0.1, batch_size = 32, epochs = 15)
```

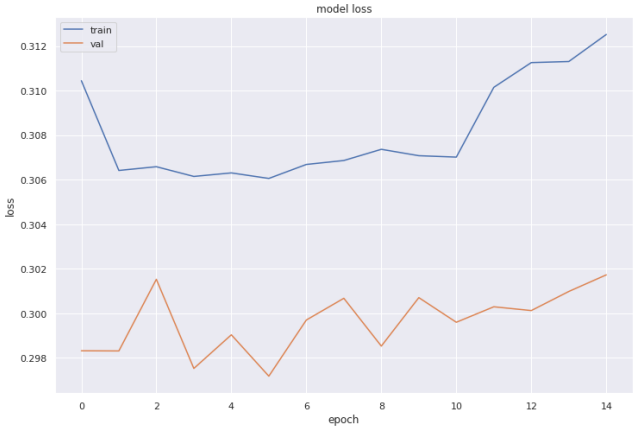
Loss and accuracy during training “main model” (include validation):

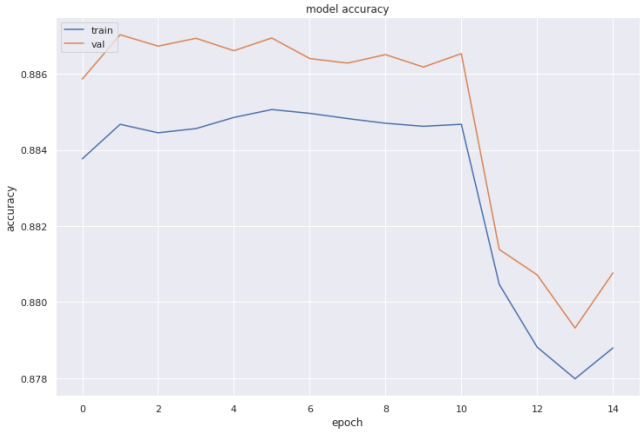


Our results for the “main model”: 89% Accuracy.

```
[[210930   4697]
 [ 27153  48563]]
Accuracy: 89%
TPR: 89%
TNR: 91%
              precision    recall  f1-score   support

    Arrested       0.89      0.98      0.93    215627
Not Arrested       0.91      0.64      0.75     75716

    accuracy                           0.89    291343
   macro avg       0.90      0.81      0.84    291343
weighted avg       0.89      0.89      0.88    291343
```

Loss and accuracy during training “BOW model” (include validation):





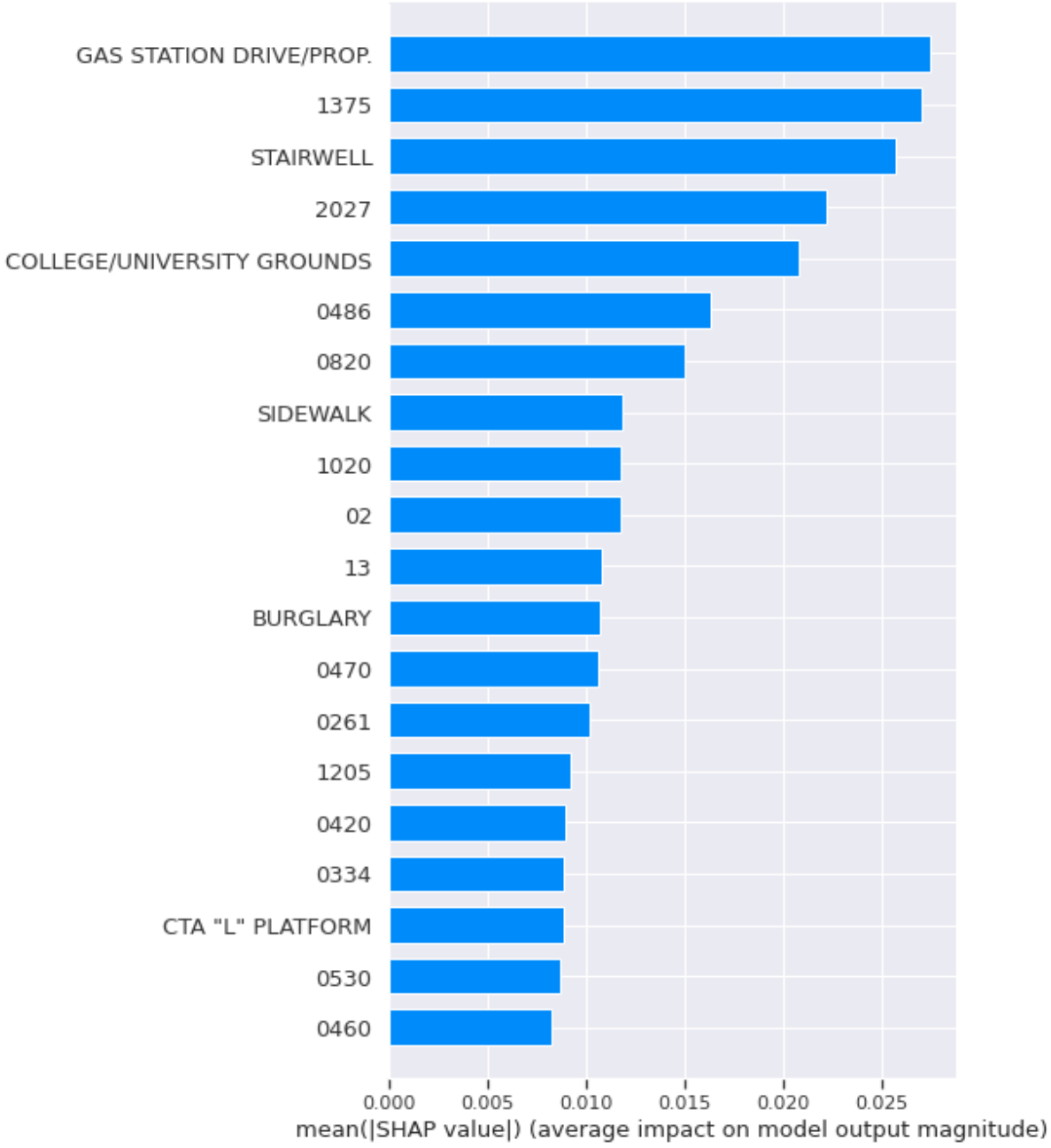
This is our results for the BOW model: 88% Accuracy.

```
[[212690   2937]
 [ 32644  43072]]
Accuracy: 88%
TPR: 87%
TNR: 94%
              precision    recall  f1-score   support

    Arrested       0.87      0.99      0.92    215627
Not Arrested       0.94      0.57      0.71     75716

    accuracy                           0.88    291343
   macro avg       0.90      0.78      0.82    291343
weighted avg       0.88      0.88      0.87    291343
```

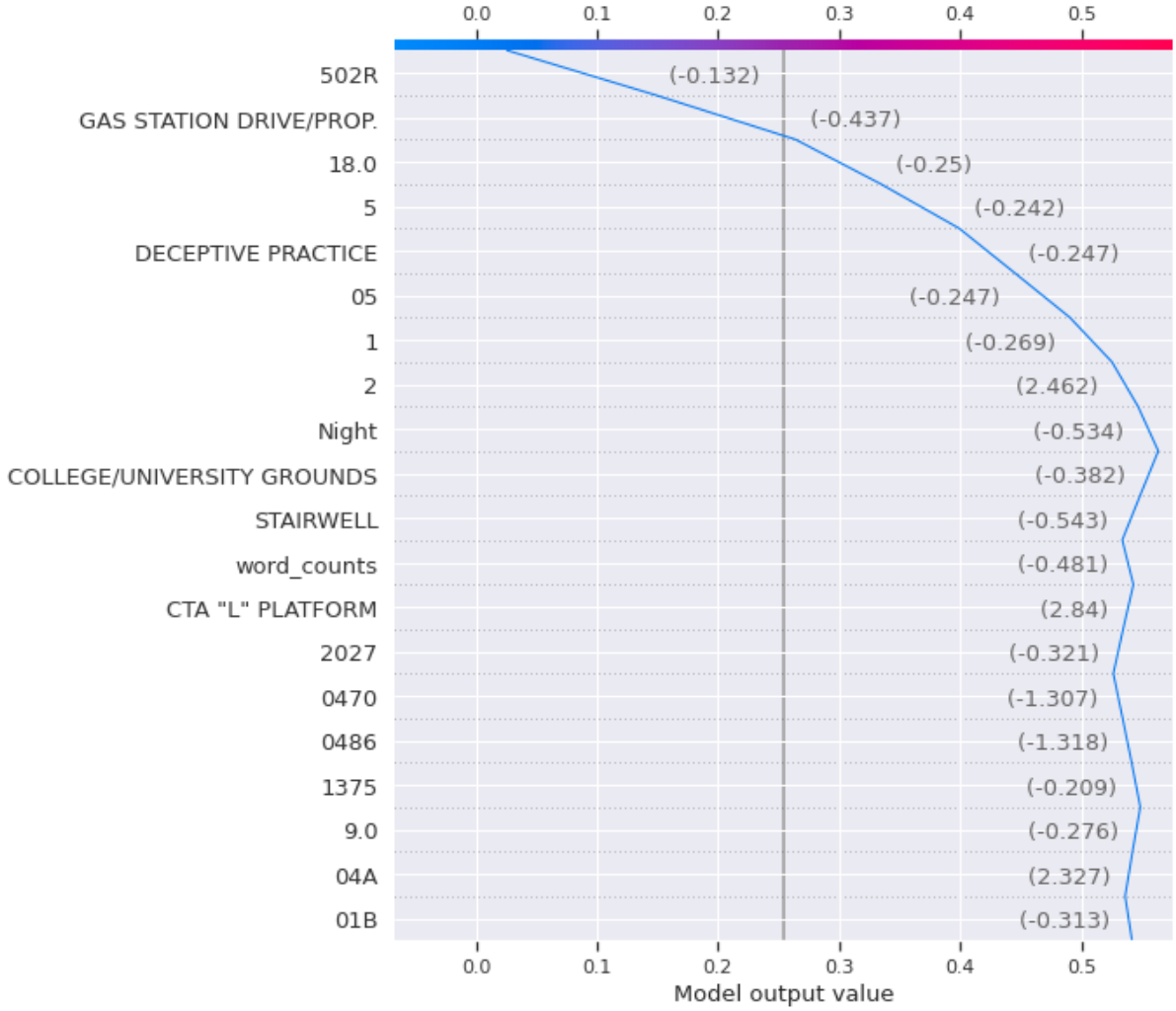
Analyze Model performance by SHAP

After training the models we choose to use SHAP to evaluate better our model and to explain the features importance.

As we can see from the summary plot below, that the significant features are the features include mainly the location where the crime was committed and the “IUCR” (the 4-digits numbers) which is the code for reporting that crime (we see name of the columns after the one-hot-encoding).



In the Decision plot we can see more features related to the time the crime happend, like the “hour” (the single/2 digit numbers) or the “part of the day” feature (Night). We can see here also the “word_count” feature which we extracted from the text report.



Insights about our analysis:

All from all, we can see that we have some significant features in our model, the location of the crime and the crime's type, the IUCR code, the date of the time, the description taken from the policeman at the time of the crime report.
We splitted our model into 2 parts the first for our features, the second part need to handle with the text of the crime's description, for that we used the "Bag-Of-Words" model which was very intresting and very learning.
We did several actions that make the model more stable relying on the analysis we made:
In addition to the analysis, we tried to clean and arrange the data as much as we could. We now understand that the data preprocessing process contributes directly to the success of the model and the drawing of our conclusions as data scienting in the "insights" stage that come after the model's test.

In the future, we would like to try new ways to extract features from text. After messing around a bit with the "Bag of words" model, we just started to understand the potential and the amount of power that language models have.