Predicting Dangerous SF Districts for Certain Crime Categories for certain day and time

Dennis Simon: 007742215, Varun Shah: 010823657, Vishweshkumar Patel: 012461371

1. Introduction

- Motivation

- The motivation behind this project was to attempt to help/assist in solving a real life problem. Thus when choosing our dataset we looked only at datasets with topics that had real life applicabilities for societies.

- Objective

- The project focuses to use past records of crime incidents in San Francisco to predict(classify) danger of specific type of crime occurrence at specific location of the district for certain day of week and time.
- The outcome of the project is to predict(classify) potential dangers/crimes (e.g. assault, battery, theft, drug use, etc.) for a respective district. The city level model to predict(classify) crime types is compared to each district level model to understand different behaviour of those models.
- The use-case provides more insights to police departments to make certain decisions to improve public safety for a respective district.

- Literature/Market review

- Referred kaggle competition, because of its relevance to predict crime categories as one of their objectives. Their predictions were done on a dataset of Los Angeles police incidents which provided different features from our dataset though.
- PREDPOL Whitepaper - PREDPOL is a predictive policing company which has designed the software that predicts the possible crime event location with the crime type in order to assist police department to prevent crime before it may occur.

2. System Design & Implementation Details

- Algorithm(s) considered/selected (and why)

- In the project we are attempting supervised learning and we decided to use Neural Networks(Vishweshkumar), XGBoost(Dennis) and Ensemble Model(Varun), The reasons are as follows.
- The reason XGBoost was used is because it is currently one of the top performing models on machine learning contests on platforms like Kaggle and it also satisfies the project requirements of being an advanced as well as state of the art algorithm.
- The reason behind using a model based on Neural Network is that in general, given a large enough dataset to generalize, Neural Network based models are more prone to give comparatively better accuracy than other Supervised Learning models. And the reason behind choosing Multilayer Perceptron Classifier (MLP) model is that it's complex enough for simple classification tasks such as one we are dealing with. The RNN and CNN model are more prefered for complex tasks such Computer Vision and NLP tasks.
- The reason behind using ensemble method is to combine the predictions of several classifiers in order to improve generalizability / robustness over a single classifier. Ensemble methods can combine several machine learning techniques into one predictive model in order to decrease variance and bias/

- Technologies & Tools used

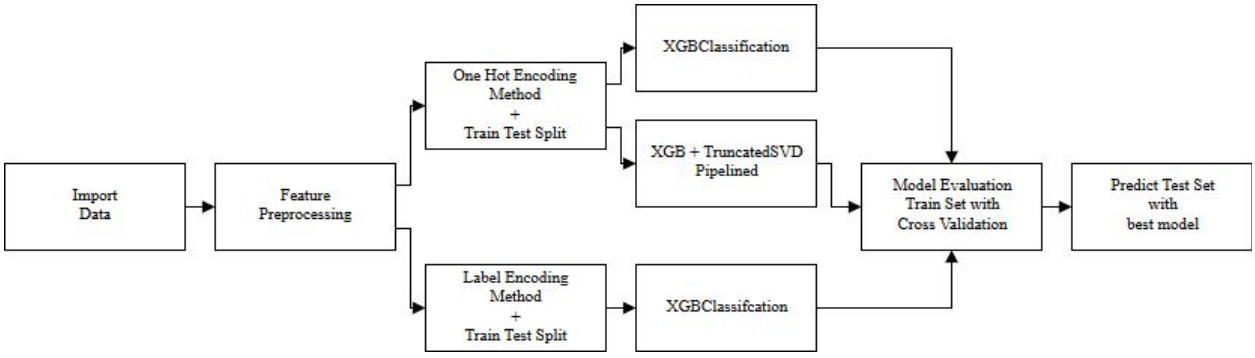
- Tools :
- AWS/EC2
    - MLPClassifier(Vishweshkumar): Deep Learning AMI Ubuntu EC2 Instance p2.xlarge utilizing 4 vCPU's as well as 61 GiB of memory
    - XGBoost(Dennis): Compute Optimized instance c5.9xlarge utilizing 36 vCPU's as well as 72 GiB of memory
    - Ensemble(Varun) : Deep Learning AMI Ubuntu EC2 Instance c5.4xlarge utilizing 16 vCPU's as well as 32 GiB of memory
- Jupyter Notebook with Kernel running on AWS EC2 Instance
- Languages and Libraries :
- Language: Python
- General Libraries: Sklearn, Pandas, Numpy, Matplotlib, Scipy, re, sodapy, seaborn, pickle
- XGBoost Specific Libraries
    - xgboost

- Architecture-Related decisions

- AWS instances were used as the systems for performing the machine learning algorithms on because normal computers do not contain the processing power required to deal with large datasets such as ours. The 36 core/72 GiB memory compute optimized EC2 instance used for XGBoost because XGBoost allows for parallelization and creating large numbers of trees.

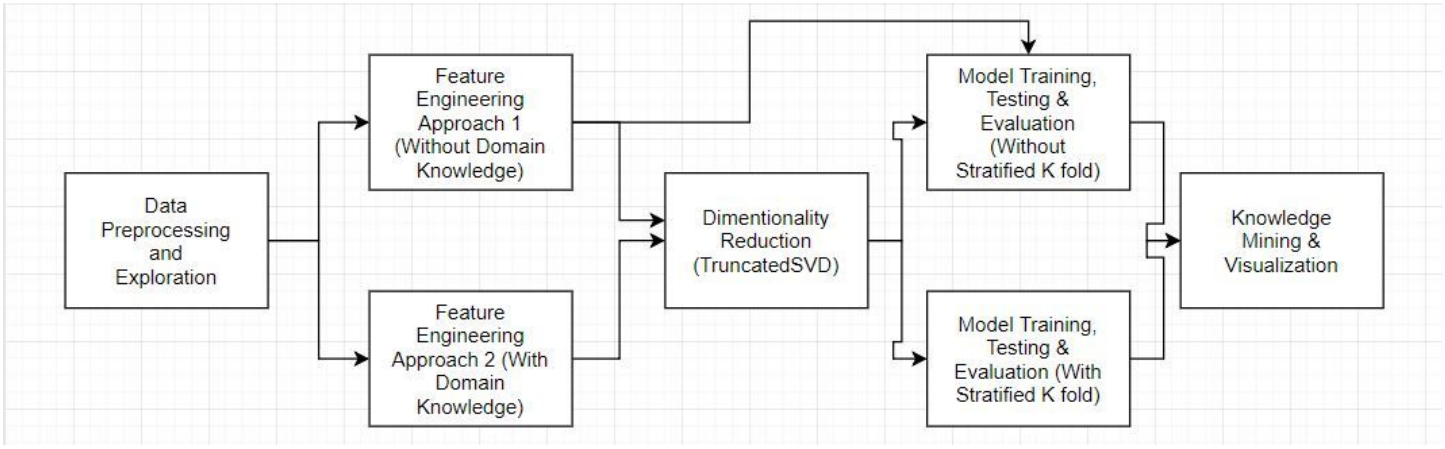
- System design/architecture/data flow

XGB Architecture for Mode

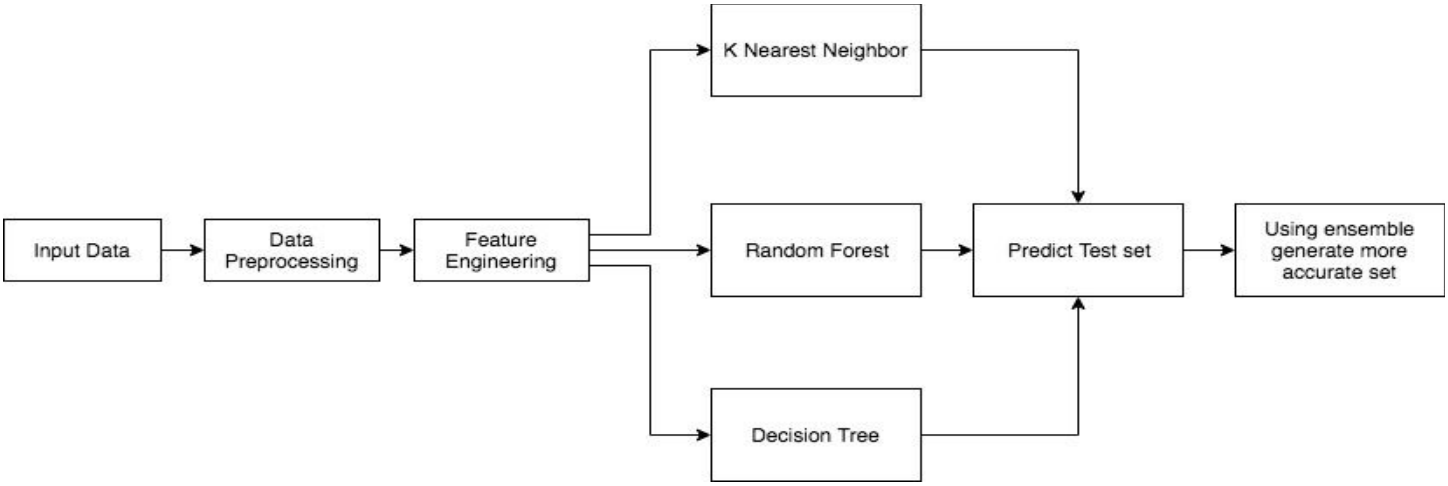


l Evaluation on Bayview District

Neural Network Architecture



Ensemble Architecture



- Component Details: Each component is explained in detail in methodologies section.

3. Experiments / Proof of concept evaluation

- Dataset(s) used (name, source, type of data, size of data, # of instances/statistics, any preprocessing performed etc.)

- Name: Police Department Incidents
- Source: DataSF
- Type of data(attributes):

Attribute	Type	Attribute Description
IncidntNum	Number	Incident number reported for a crime
Category	Text	Category of a crime
Descript	Text	A short description of crime incident
DayOfWeek	Text	Day, when crime happened

Date	Date	Date, when crime happened
Time	Time	Time, in between “00:00” to “23:59”
PdDistrict	Text	Police department district
Resolution	Text	Police action(s) for respective crime
Address	Text	Apt and Street address where crime happened
X	Number	Longitude
Y	Number	Latitude
Location	Location	Location
Pdid	Number	Unique Identifier for use in update and insert operations

- Size of Data: 13 Features/Columns, 2,206,399 rows
- Group Data Preprocessing: 1)Moved Category column to separate output array and removed from features, 2)Removed incident number from features because only used for police department purposes, 3)Removed pdid because used for police department identifier operations, 4)Removed location because we have x and y coordinates already(location is x, y concatenated), 5)Removed description as it describes/pertains to category thus unusable, 6) Removed pddistrict for district models as they’re already separated by district, 7) Convert date(dd/mm/yy) to three columns of day, month and year, 8) Convert hour from exact(e.g. 22:35) to rounded(e.g. 22) to generalize.

- Methodology followed (e.g. n fold, cross validation, number of folds, size of training/test set etc.) (as applicable)

XGBoost (Sections refer to pipeline found in Architecture Section):

- Feature Preprocessing & Split: XGBoost uses numeric input/outputs so need to transform categories from string to numeric. Done once using LabelEncoder which transforms to one ordinal column of integers and in second case using one hot encoding to transform into many columns of binary values. Both options tested because even if feature doesn’t have ordinal relationships sometimes XGBoost can find connections between them. Stored in CSR sparse matrix because one hot encoding produces sparse columns. Data split into train-test split(test size of 20%) to evaluate methods/tune hyperparameters on train set and use best on test set. Note: I kept resolution as feature contrary to teammates as several similar implementations(e.g. LACrime EDA) utilized resolution as a feature as it is just a status in case(e.g. Booked, etc.).
- Modelling: First model tested was XGB on label encoded set, second model tested was XGB on one hot encoded set and third model tested was XGB+truncated SVD in a pipeline on one hot encoded set. Pipelined because if tSVD done outside then cross validation folds aren’t separated. All models evaluated with nthread = 36 to optimize by using XGB multi-core parallelization, n_jobs = -1 to utilize all available cores in parameter search.
- Model Evaluation: All 3 models evaluated in randomized search CVs because randomized search can tune parameters effectively while also being less compute/time intensive than gridsearch, used 3 iterations because

time did not permit more(long computation per iteration). Stratified 3 fold cross validation used, because there are large number of categories as well as imbalanced dataset and stratified kfold enforces similar class distributions in each fold to mitigate these issues. 3 Folds because more folds increases computation time by hour(s) and standard deviation between parameter tunes was low with this still. Parameters searched/tuned: n_estimations(120-480) because more estimators increases time required by hours, max_depth(2-8) because higher depth increases overfit and 3-10 is generally accepted guideline, learning_rate(.01-.08) because low rate gets optimum steps but increases computation time, colsample_bytree(.3-.7) to perform stochastic GB for XGB by using random sample of features with a max of .7 to ensure some sampling done to prevent overfitting. For SVD number of components randomly varied from 5 to 1000 to examine how sparse features affected in importance.

- Final Steps: Apply best method with tuned parameters to test set and save f1 score for district comparison. Model evaluation only done on Bayview, best model(XGB one hot encoded) was applied to every district but tuned for each individual district because process is computation/time intensive. The city wide model also used the district as a feature to see the effect on scoring as compared to the individual districts which obviously did not have it.

Neural Network (MLP Classifier)

- Explored problem in two ways: 1) Direct approach without domain knowledge on available features, after achieving very less accuracy(~20%) thought to use domain knowledge to solve problem, 2) Approach considering domain knowledge and modification of available features. In second approach using FBI's crime classification categories generalized crime categories in two categories: Index(More Serious) crime and Non-index(Less serious) crime and understood and implemented concept of hotspots from NCJRS to only consider certain addresses where crime occurs more frequently, which can help to efficiently allocate resources for a certain district. - Followed object oriented approach for ease to handle different district and to create model for each district and for city. Used four function to represent each phase of data science life cycle namely A) Data Preprocessing and Exploration: Followed same data preprocessing in both of the approaches in following order. 1) Removed 'resolution', 'descript' features, as they can't be known beforehand. 2) Removed 'pdid', 'incidntnum' as they are not relevant. 3) Removed 'location', 'x', 'y' features as already considering 'address' feature that provides needed granularity of the location where crime happened.4) Removed null values and duplicate values for remaining data points. 5) Segregated data points based on their district values and stored in different data frame. B) Feature Engineering(Extraction, Transformation, Reduction): *For direct approach*, 1) Extracted values of 'day', 'month' from 'date' and 'hour' from 'time' feature, and removed 'date' & 'time' feature. 2) Segregated features ('address', 'month', 'day', 'dayofweek', 'hour') and target('category') in different data frames. 3) All input features were categorical so converted them in to binary features (same as one-hot encoding but using custom logic) and due to it's sparsity and to handle them efficiently converted them all into a Compressed Sparse Row matrix. 4) Applied 'TruncatedSVD' dimensionality reduction technique on CSR Matrix and plotted explained variance graph to find out inflection point and consider only high variance components. *Considering domain knowledge approach,*After applying first two steps, 1) Added a new column 'Crime_severity' having two unique values 'Index crime' and 'Non Index' crime based on 'crime category' and 'descript' column and using FBI standards (thus, domain knowledge), 2) Decided hotspot criteria as "An

address is a hotspot if there are more than average number of crime incidents occurred at that same address for same 'hour', 'dayofweek', 'month' ". *Filtered data points and only kept hotspot addresses because they need more attention compared to other addresses and predicting severity of crime type at hotspot for certain day, hour, month can help Police Department's to optimize their resource allocation in relevant hotspots*. Apart from that followed similar 3rd and 4th steps after that. C) Model Training, Testing and Evaluation: 1) Used stratified K fold technique with K=2, to handle imbalanced data if any present. Kept K=2 due to limitation of computation resources (as K increases, MLP classifier is executed for K times and for all of them considering different parameter tuning such as hidden layers and number of nodes in one layer, etc. become impossible). 2) Tried to perform Randomized Search with different parameter range but again faced same problem - limitation of computation resources, So chose parameters one by one experimenting with different values of solver function, hidden layers and nodes, alpha, max_iter(epochs) and learning rate. 3) Prevented overfitting through appropriate value of alpha (regularization parameter) and employing bias vs variance tradeoff, 4) Stored best training & testing f1 scores for different districts and city for comparison and visualization purpose. D) Knowledge Mining and Visualization: Plotted Barcharts to compare performance of different approaches, related graphs are provided in next section.

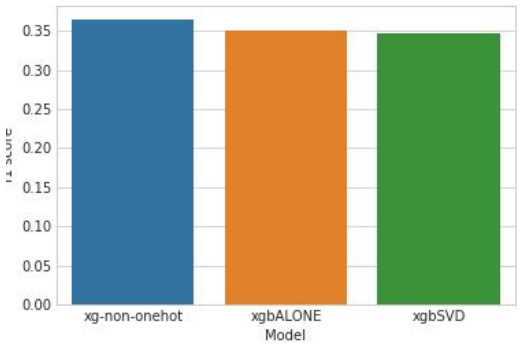
Ensemble Model

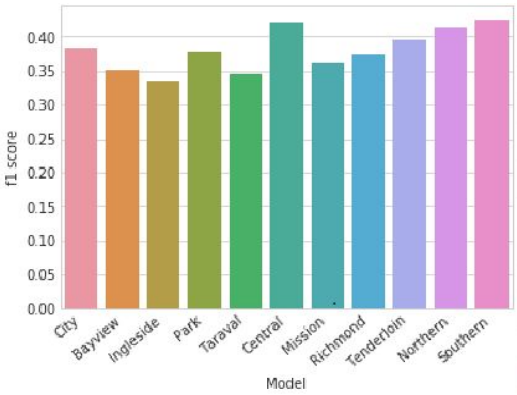
- Data Preprocessing and Exploration: Followed data processing following steps: (1) Sort data by date and time to have it in chronological order. (2) With one approach considered all categories and for second approach considered (crime type guideline) to categorize current categories to further 2 categories: index crime and non-index crime (3) removed irrelevant columns like 'incidntnum','pdid'
- Feature Engineering(Extraction, Transformation, Reduction): (1) created new columns day and month from date and hour from time and removed date and time columns (2) Removed irrelevant columns:,'resolution', 'descript' and 'location'. (3) converted all string data to lowercase and applied label encoding to convert string data to integer (4) For location, considered longitude and latitude till 3 decimal points in order to consider specific block at specific street (5) created new column severity which has two categories: index crime and non-index crime
- Model Training, Testing and Evaluation: Using train_test_split data split into train and test. Ensemble with following classifiers: K-Nearest Neighbor, ADABoost, Random Forest and Decision Tree. Performed parameter tuning using GridSearch and in K-Nearest Neighbor, due to memory error parameter tuning was performed manually. While manually tuning parameters, best performance achieved by n_neighbors 3 as there are 7 features and using minkowski metric. Using ensemble method, it seems that all models provide near about same accuracy for all the classifiers.
- Knowledge Mining and Visualization:

Plotted Barcharts to compare performance of different approaches, related graphs are provided in next section.

- Graphs showing different parameters/algorithms evaluated in a comparative manner, along with some supportive text. (as applicable)

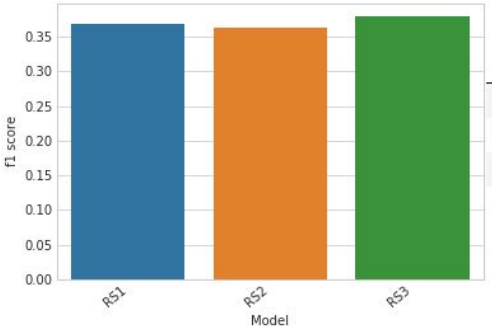
XGBoost:





	model	f1
0	City	0.382020
1	Bayview	0.352562
2	Ingleside	0.335201
3	Park	0.377947
4	Taraval	0.346276
5	Central	0.420972
6	Mission	0.361464
7	Richmond	0.374957
8	Tenderloin	0.396360
9	Northern	0.413667
10	Southern	0.424929

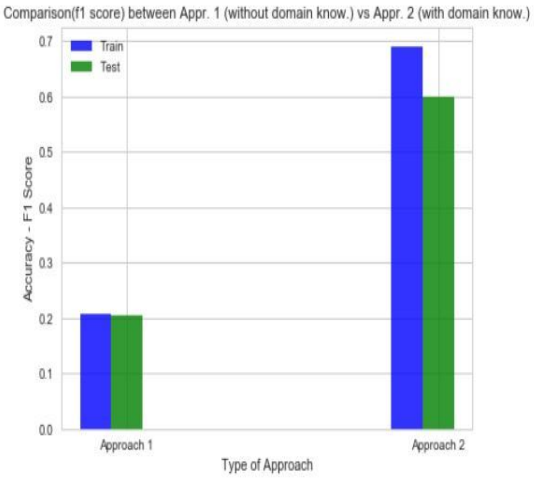
Figure(Left). Comparison of XGB modelling processes used on Bayview District
Figure(Right). Displaying the f1 scores of XGB on each district as well as the whole city

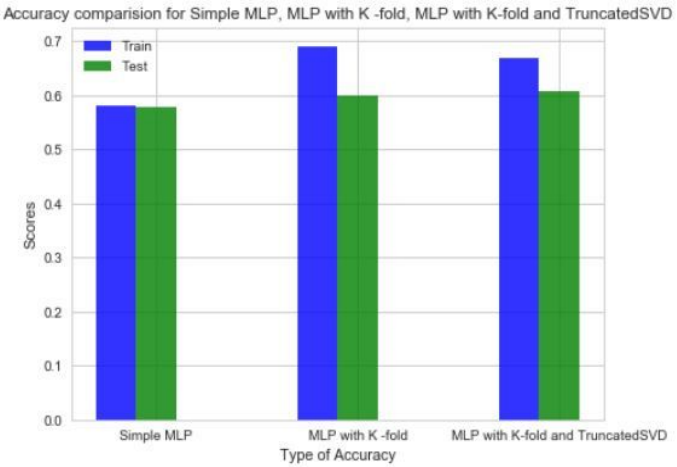


	Model Name	Scores	col_samplesbytree	learning_rates	max_depths	# estimators
0	RS1	0.368172	0.798051	0.074796	4	240
1	RS2	0.361924	0.527669	0.061418	4	120
2	RS3	0.379332	0.999938	0.080407	8	240

Figure. F1 scores of different XGB on City Data with different parameters in Cross Validation

Neural Network (MLP Classifier)





Figure(Left). Comparison(f1 score) between Appr. 1 (without domain know.) vs Appr. 2 (with domain know.)
Figure(Right). Comparison(f1 score) of different combinations for MLP modelling process on Bayview district

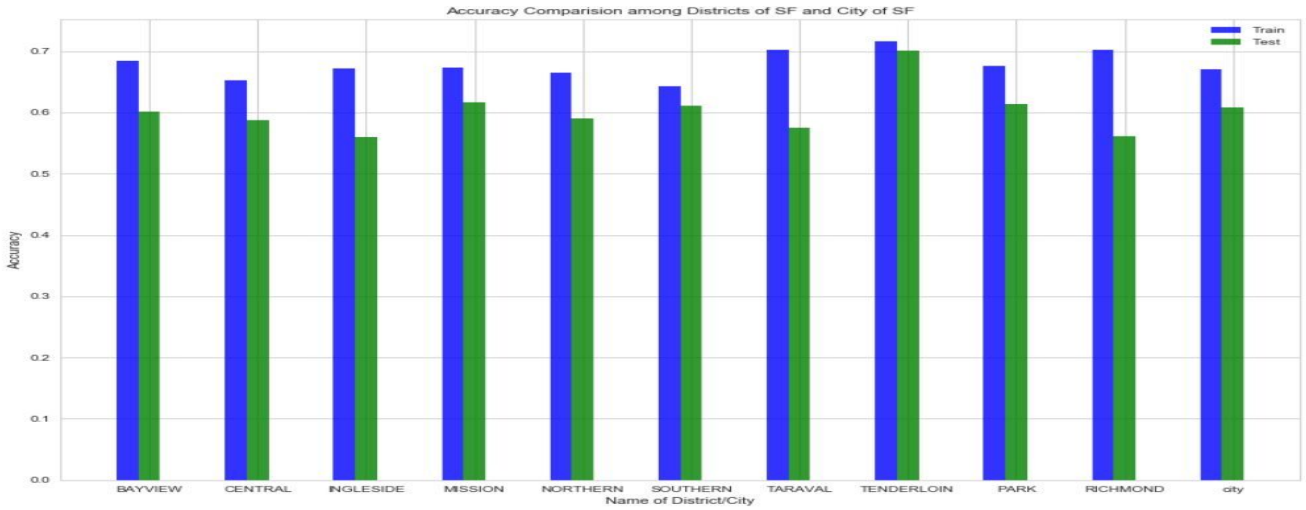
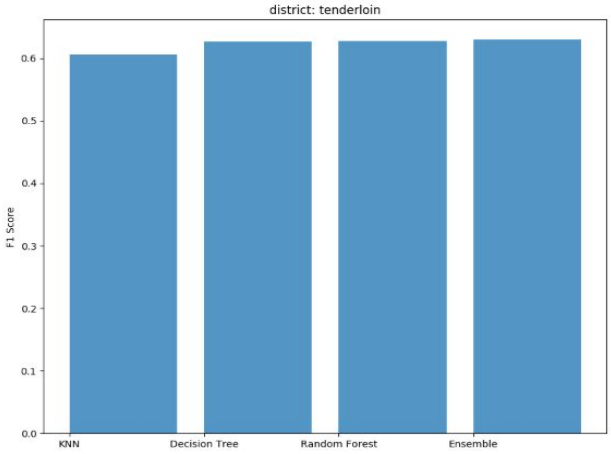
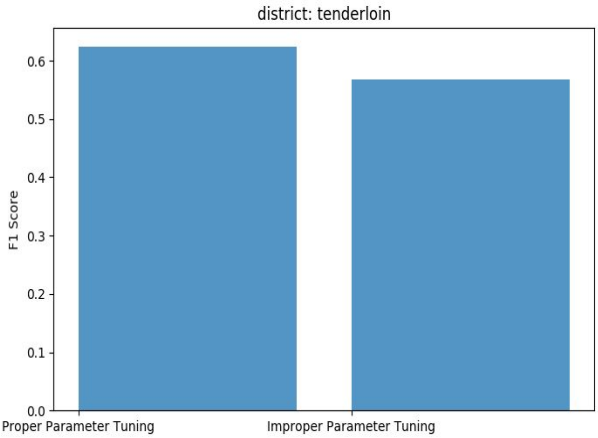


Figure. Train & Test accuracy comparison(f1 score) among districts of SF and city of SF(last bar)

Ensemble



fig(1)



fig(2)

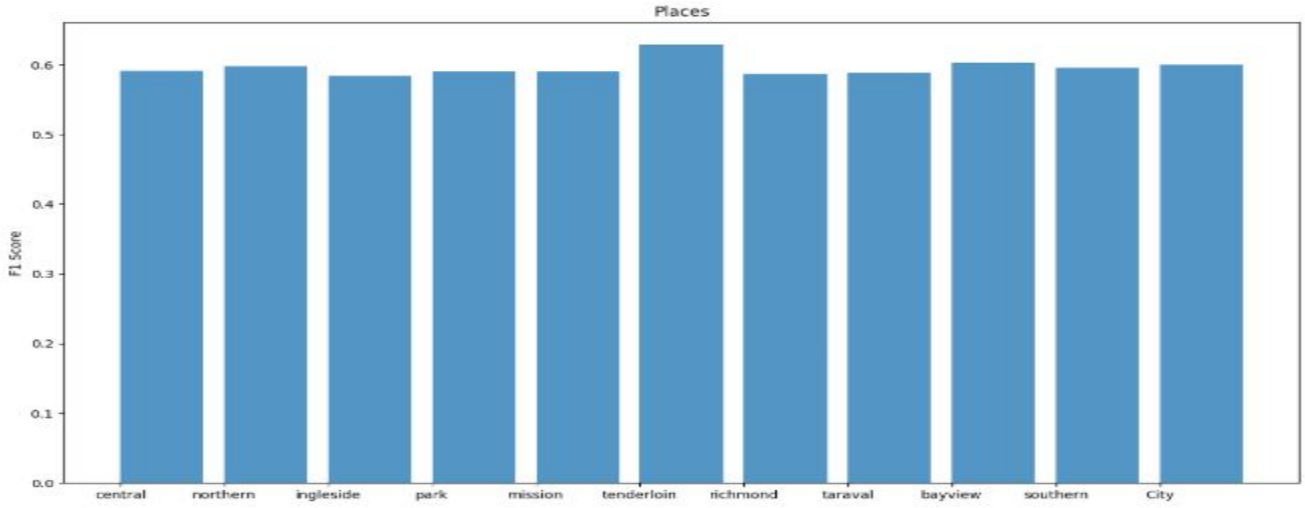


Fig (3)

- Analysis of results

XGBoost

All three modelling options performed similarly but using one hot encoded dataset took around half of the time when compared to using label encoded dataset. XGB with truncated SVD was also slower than without and performed slightly worse. Adding district feature for the city wide search did not significantly improve the f1 score as compared to the district wide searches. F1 scores hovered around 34-40%, I attribute this to not having enough relevant features for classifying 39 different categories especially since we had to remove description which likely would've had the most importance for classifying. Note: partners categorized 39 categories into two categories representing danger of categories but decided I wanted to evaluate my ability to categorize specific categories.

Neural Network (MLP Classifier)

From first figure, we can analyse that Approach 2 (With domain knowledge) is far better than Approach 1 (Without domain knowledge), and Approach 2 also focuses on hotspots thus it narrow downs area to be considered for resource allocation and only predicts severity of crime for those addresses. From second figure, we can analyse that MLP with K fold and MLP with K fold and Truncated SVD nearly gives similar performance the reason may be associated with sparsity of data. From third figure, we can analyse that f1 score for district varies from 56% to 72% (both for train and test split) and for city it also falls in same range, thus city level model and district level model for Approach 2 has not much variation to exhibit.

Ensemble

From first image, it can be analyzed that performing ensemble of KNN, Decision tree and Random forest gives equal or better performance. In 2nd Image. It is shown that proper parameter tuning is very important. High accuracy graph of 0.62 has max_depth [None, 50, 70] where as low f1 score of 0.56 has max_depth [None, 5, 10]. From this it can be deduced that higher depth provides more accurate prediction. 3rd figure depicts san francisco district and city prediction f1 score using ensembling.

4. Discussion & Conclusions

- Decisions made

- Don't use description feature as it provides info related to category.
- XGB:Use 36 core EC2 because 16 core c5.4xlarge had memory deficiency, Keep all 39 categories, Don't use GPU Boosting as issues with sklearn implementation, No gamma/regularization tuning per Owen Zhang(top kaggle submitter) tips.
- MLPClassifier (Vish): To propose an effective solution for resource allocation in district and city, referred domain knowledge (generalized crime categories and hotspot analysis) and then reshaped data and solution in that context to deliver valuable output, used CSR matrix to efficiently handle sparse data.
- Ensemble: Decided to use longitude and latitude as a feature by using it till 3rd decimal point, as it is meaningful feature which depicts exact location of crime, used EC2 C5.9xlarge instance

- Difficulties faced

- XGB:One hot encoding city data led to memory issues w/ 36 and 72 EC2 cores, Lack of relevant features to model on causing low f1 scores, High computation times, hours per district at times even with parallelization.
- MLPClassifier(Vish): More resource needed to implement grid and/or randomized search on complex algorithm such as Neural Network, applied domain knowledge but

- Ensemble: Lack of domain knowledge. Also, having 39 categories and lack of features to predict category for at specific day, time in respective district. With grid search with K-Nearest Neighbor memory error on 36 cores EC2 Instance

- Things that worked

- XGB:Changing OHE to output sparse matrix decreased memory requirements
- MLPClassifier(Vish): After applying domain knowledge for generalizing crime and considering hotspots accuracy improved drastically, and it was more relevant solution as there are considerably less addresses(locations) to consider while allocating resources, which are also prime locations of crime, for which prediction is made.
- Ensemble: considering longitude and latitude till 3 decimal points, classifiers gave really good results. And ensembling RandomForest, KNN, ADABoost all three provide near about equal f1 score

- Things that didn't work well

- XGB: Randomized search over more iterations would tune better
- MLPClassifier(Vish): Grid search & Randomized search over different range for different parameters may provide more accuracy
- Ensemble: Grid search for K-Nearest Neighbor to tune parameters gave memory error, address could be generalized by block, street and no of crimes at respective place, High computing system to handle such a large dataset

- Conclusion

- Data Preprocessing and exploration is more important than any other steps in data science life cycle - as we say 'Garbage in - Garbage out'.
- Help from Domain expert could have provided a different point of view to solve a problem, may be more easier, effective and efficient way could have been found.
- As each of us followed different approach to solve problem our models are not directly comparable but in general ensemble of XGBoost, MLP and Ensemble Model may provide more precise accuracy than individuals.

5. Project Plan / Task Distribution

- Who was assigned to what task

- Individually searched for potential data sets and problems that can be solved using them, met together to evaluate them.
- General data preprocessing was to done by all of us together, further model specific preprocessing done individually.
    - XGBoost Model: Dennis Simmon
    - Neural Network Model (MLP Classifier): Vishweshkumar Patel
    - Ensemble Model: Varun Shah
- Report and slides done together but individually for specific models.

- Who ended up doing what task (justify as applicable)

- We each ended up doing each task and model as originally assigned.