Crime scale prediction

Amelia Krokosz, Cezary Dymicki, Paweł Umiński

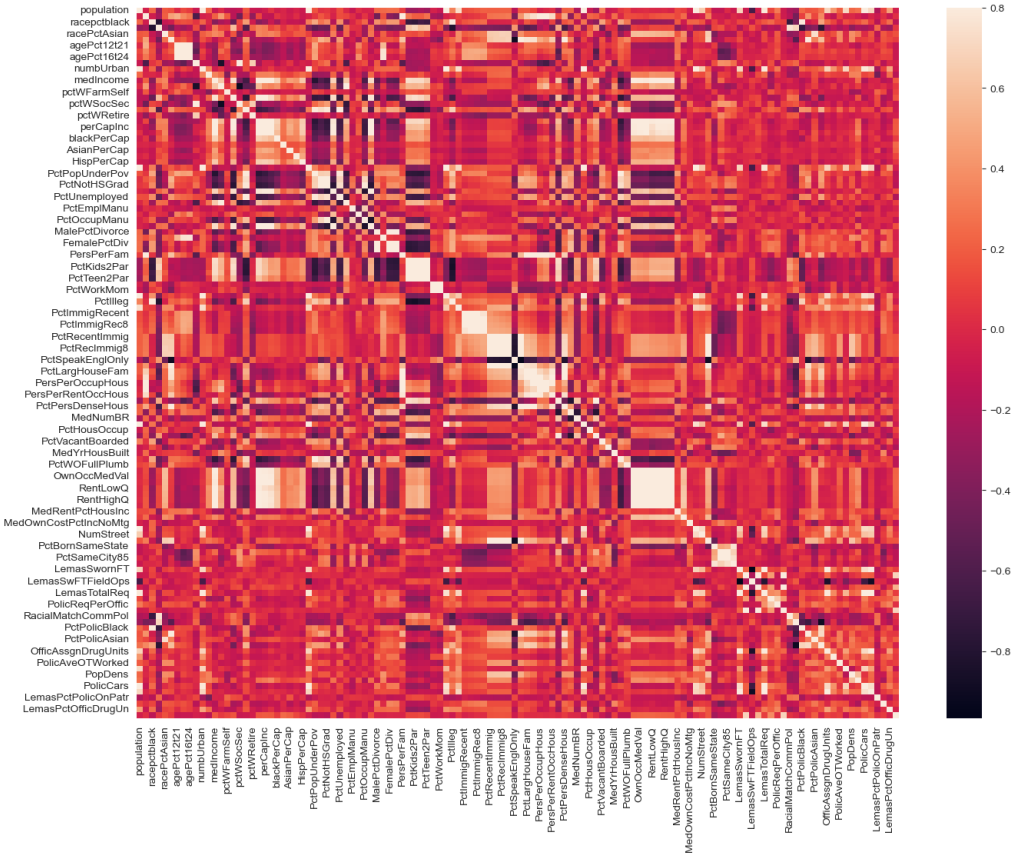
January 27, 2024

1 Introduction

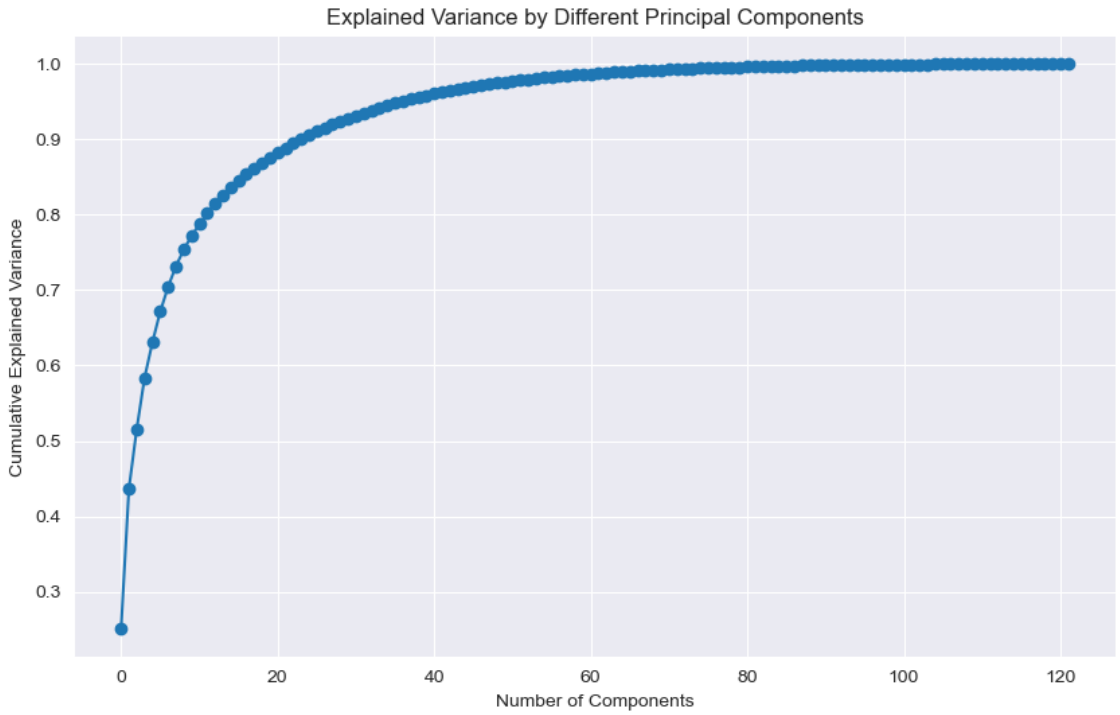
The aim of this research project is to develop a predictive model for estimating crime rates in a new region based on its demographic data. This report presents the methodology, data preprocessing, model evaluation, experimental results, and conclusions of the study.

2 Dataset overview

The dataset consists of 128 attributes and 1994 records. 5 of the features are non-predictive, 1 is the target column. There are 21 columns which have missing values, there is 1 record missing a single value in the 'OtherPerCap' column and there are 1675 records missing values in the other 20 columns.



A correlation matrix depicting linear relationships between variables



A diagram showing the relationship between number of PCA components and the variance explained by them.

3 Data Preprocessing

3.1 Handling missing values

We approached the problem by utilising the Multivariate Imputation by Chained Equation (MICE for short). It's an informative method for dealing with missing data. It works by combining multiple results from different imputation methods to produce a complete dataset. First it changes all missing values to placeholder values (for example the mean of the column), then for the first column that has missing values it changes the plceholders back to missing values and then applies linear regression this particular column to predict its values, and it does so for all the columns. It is implemented in python in the package IterativeImputer from the sklearn library.

Another approach we used was using Autoencoders to impute the missing data. They are neural netowrks which first reduce the dimentionality of a dataset (Encode) trying to capture the most relevant data and then decode it, aiming to be as accurate as possible. It is first trained with a mask loss function which is only calculated on the not missing data, the network initially fills the missing data with trivial imputation like the mean and then learns to observe better patterns and augment the data more accurately. Its is not directly implemented in python, so we used tensorflow's layers Dense and Drouput, optimizer Adam, keras' regularizer l1l2 and implemented early stopping to prevent the model from overfitting.

Our final 6 datasets are: Mice, Autoencoded, MiceTop30Correlated, AutoencodedTop30Correlated, MicePCA, AutoencodedPCA. The MICE imputation seems to perform the best, judging by all model's performance, which is mentioned later in the paper.

3.2 Selecting features of significance

In our base dataset, we dropped the columns which are non-predictive. We used PCA (Principal Component Analysis) to create a dataset with reduced dimension of our dataset from 122 to 30. PCA works by creating new features that capture most of the variability of the data, hence these features

are called Principal Components. We performed PCA, with number of components equal to 30, to the data augmented by both MICE and by AutoEncoder.

Also we attempted to find the 30 features most linearly correlated to the target variable, taking into consideration the lowest, negative correlations and the highest, positive ones, by calculating their absolute values and sorting descendingly.

Unfortunately, we observed that dimensionality reduction did not always help in achieving lower losses/

4 Model Information

4.1 Optimal parameters

In order to select parameters that performed best, we used Hyperopt. It works by first defining a hyperparameter space with parameters and their possible values, choosing function which we wish to optimize and then an optimization algorithm, TPE or RandomSearch. In our case, its Tree-structured Parzen Estimator (TPE). It uses a Parzen tree model to adaptively choose the best new parameters to check. It's based upon Bayesian optimisation, which works by estimating a function in one point, because estimating it in all points would be very computationally expensive, and then narrowing down the best possible parameters, picking a new point and doing the same until satisfaction.

4.2 The Models

We tested out different models (LinearRegression, RANSACLinearREgression, Lasso, Ridge, ElasticNet, MLPRegressor, RandomForest, XGBRegressor, LightGMB, Catboost) with each dataset.

We noticed an increase in R^2 in LinearRegression and RANSACRegression after performing PCA.

model	**Parameters**
RANSACRegressor	('loss': 'squared_error', 'max_trials': 200, 'min_samples': 0.1, 'residual_threshold': 5)
Lasso	('alpha': 0.001, 'max_iter': 5000, 'tol': 0.001)
Ridge	('alpha': 10, 'solver': 'sag')
ElasticNetCV	('l1_ratio': 0.2, 'max_iter': 1000)
SVR	('C': 0.1, 'epsilon': 0.01, 'gamma': 'scale', 'kernel': 'rbf')
RandomForestRegressor	('max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100)
XGBRegressor	('colsample_bytree': 0.9, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5, 'n_estimators': 100, 'subsample': 0.8)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.5008017965932141, 'learning_rate': 0.06797634227648623, 'max_depth': 15, 'min_child_weight': 0.5894881671454538, 'n_estimators': 100, 'num_leaves': 20, 'subsample': 0.5262867509552965, 'verbose': -1, 'verbose_eval': -1)
CatBoostRegressor	('bagging_temperature': 0.10071749981977751, 'border_count': 64, 'colsample_bylevel': 0.7827790429887673, 'custom_metric': ('RMSE',), 'depth': 6, 'eval_metric': 'RMSE', 'iterations': 1500, 'l2_leaf_reg': 4.34732759050278, 'learning_rate': 0.05910294867691535, 'min_child_samples': 5, 'random_strength': 0.9995518556088577, 'subsample': 0.9600199577334129, 'verbose': False)

Table 1: Best Parameters on MICE

model	**Parameters**
RANSACRegressor	('loss': 'squared_error', 'max_trials': 50, 'min_samples': 0.1, 'residual_threshold': 5)
Lasso	('alpha': 0.001, 'max_iter': 10000, 'tol': 0.001)
Ridge	('alpha': 10, 'solver': 'sag')
ElasticNetCV	('l1_ratio': 0.5, 'max_iter': 5000)
SVR	('C': 0.1, 'epsilon': 0.01, 'gamma': 'scale', 'kernel': 'linear')
RandomForestRegressor	('max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100)
XGBRegressor	('colsample_bytree': 0.9, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3, 'n_estimators': 100, 'subsample': 0.8)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.5696921403896573, 'learning_rate': 0.02466697927701829, 'max_depth': 20, 'min_child_weight': 0.6648216551625716, 'n_estimators': 200, 'num_leaves': 20, 'subsample': 0.9449980581935515, 'verbose': -1, 'verbose_eval': -1)
CatBoostRegressor	('bagging_temperature': 0.30838453525960546, 'border_count': 64, 'colsample_bylevel': 0.6777070706161952, 'custom_metric': ('RMSE',), 'depth': 6, 'eval_metric': 'RMSE', 'iterations': 1000, 'l2_leaf_reg': 8.83450447340915, 'learning_rate': 0.08306210135428686, 'min_child_samples': 20, 'random_strength': 0.6941319681418067, 'subsample': 0.5452260251642657, 'verbose': False)

Table 2: Best Parameters on AUTOENCODER

model	Parameters
RANSACRegressor	('loss': 'squared_error', 'max_trials': 50, 'min_samples': 0.5, 'residual_threshold': 5)
Lasso	('alpha': 0.001, 'max_iter': 1000, 'tol': 0.0001)
Ridge	('alpha': 1, 'solver': 'sag')
ElasticNetCV	('l1_ratio': 0.2, 'max_iter': 1000)
SVR	('C': 10, 'epsilon': 0.01, 'gamma': 'scale', 'kernel': 'poly')
RandomForestRegressor	('max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 100)
XGBRegressor	('colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5, 'n_estimators': 100, 'subsample': 0.8)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.550111669001569, 'learning_rate': 0.020458341278520118, 'max_depth': 5, 'min_child_weight': 0.3611053821274013, 'n_estimators': 200, 'num_leaves': 20, 'subsample': 0.653529197257181, 'verbose': -1, 'verbose_eval': -1)
CatBoostRegressor	('bagging_temperature': 0.11652892545084761, 'border_count': 128, 'colsample_bylevel': 0.815173262002999, 'custom_metric': ('RMSE',), 'depth': 4, 'eval_metric': 'RMSE', 'iterations': 1500, 'l2_leaf_reg': 4.4007570627658135, 'learning_rate': 0.20450513218531394, 'min_child_samples': 10, 'random_strength': 0.5679192871914768, 'subsample': 0.7591056908533966, 'verbose': False)

Table 3: Best Parameters on top30mice

model	Parameters
RANSACRegressor	('loss': 'absolute_error', 'max_trials': 50, 'min_samples': 0.1, 'residual_threshold': 10)
Lasso	('alpha': 0.001, 'max_iter': 10000, 'tol': 0.0001)
Ridge	('alpha': 10, 'solver': 'sparse_cg')
ElasticNetCV	('l1_ratio': 0.2, 'max_iter': 5000)
SVR	('C': 100, 'epsilon': 0.1, 'gamma': 'auto', 'kernel': 'linear')
RandomForestRegressor	('max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 100)
XGBRegressor	('colsample_bytree': 0.9, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5, 'n_estimators': 100, 'subsample': 0.7)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.8339634390510092, 'learning_rate': 0.038033975030615674, 'max_depth': 5, 'min_child_weight': 0.144850930760892, 'n_estimators': 200, 'num_leaves': 20, 'subsample': 0.8781803371536989, 'verbose': -1, 'verbose_eval': -1)
CatBoostRegressor	('bagging_temperature': 0.7850729235847319, 'border_count': 256, 'colsample_bylevel': 0.8267621602468271, 'custom_metric': ('RMSE',), 'depth': 4, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 8.23885365687151, 'learning_rate': 0.14668278918725877, 'min_child_samples': 20, 'random_strength': 0.6607737340437783, 'subsample': 0.8264923910301799, 'verbose': False)

Table 4: Best Parameters on top30auto

model	Parameters
RANSACRegressor	('loss': 'absolute_error', 'max_trials': 200, 'min_samples': 0.5, 'residual_threshold': 15)
Lasso	('alpha': 0.001, 'max_iter': 5000, 'tol': 0.0001)
Ridge	('alpha': 0.1, 'solver': 'cholesky')
ElasticNetCV	('l1_ratio': 0.2, 'max_iter': 5000)
SVR	('C': 0.1, 'epsilon': 0.1, 'gamma': 'auto', 'kernel': 'linear')
RandomForestRegressor	('max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100)
XGBRegressor	('colsample_bytree': 0.9, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3, 'n_estimators': 100, 'subsample': 0.7)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.5003893423405277, 'learning_rate': 0.024295119786774608, 'max_depth': 5, 'min_child_weight': 0.4447225790619702, 'n_estimators': 300, 'num_leaves': 30, 'subsample': 0.5784054305536382)
CatBoostRegressor	('bagging_temperature': 0.46298775456660424, 'border_count': 64, 'colsample_bylevel': 0.9031978778498035, 'custom_metric': ('RMSE',), 'depth': 6, 'eval_metric': 'RMSE', 'iterations': 1500, 'l2_leaf_reg': 5.165127860800862, 'learning_rate': 0.043520680717204954, 'min_child_samples': 5, 'random_strength': 0.1344647351072205, 'subsample': 0.7366653103831214, 'verbose': False)

Table 5: Best Parameters on PCA_MICE

model	Parameters
RANSACRegressor	('loss': 'squared_error', 'max_trials': 50, 'min_samples': 0.5, 'residual_threshold': 15)
Lasso	('alpha': 0.001, 'max_iter': 10000, 'tol': 0.0001)
Ridge	('alpha': 10, 'solver': 'lsqr')
ElasticNetCV	('l1_ratio': 0.8, 'max_iter': 5000)
SVR	('C': 1, 'epsilon': 0.01, 'gamma': 'scale', 'kernel': 'rbf')
RandomForestRegressor	('max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200)
XGBRegressor	('colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5, 'n_estimators': 200, 'subsample': 0.9)
LinearRegression	('.': '.')
lightgbm	('colsample_bytree': 0.5292770577347236, 'learning_rate': 0.030758656449751016, 'max_depth': 5, 'min_child_weight': 0.6860304924075082, 'n_estimators': 300, 'num_leaves': 30, 'subsample': 0.5438644771653159, 'verbose': -1, 'verbose_eval': -1)
CatBoostRegressor	('bagging_temperature': 0.6529896870123736, 'border_count': 128, 'colsample_bylevel': 0.8410343025672599, 'custom_metric': ('RMSE',), 'depth': 4, 'eval_metric': 'RMSE', 'iterations': 1000, 'l2_leaf_reg': 1.7337590660508309, 'learning_rate': 0.1216865653578193, 'min_child_samples': 5, 'random_strength': 0.05492699697276038, 'subsample': 0.6301087422502245, 'verbose': False)

Table 6: Best Parameters on PCA_AUTO

5 Model Evaluation

The metrics we used to evaluate our models were the Mean Squared Error and The Coefficient Of Determination

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$R^2 = 1 - \frac{RSS}{TSS}$$

Where:

RSS: sum of squares of residuals

TSS: total sum of squares

6 Experimental Results

After training all models, we achieved the following results with RandomForest performed on mice data, which performed best:

$$MSE \approx 0.01$$

$$R^2 \approx 0.67$$

7 Summary and Conclusions

In conclusion, our research aimed to develop a predictive model for estimating crime rates in a new region based on its demographic data. Through rigorous experimentation and analysis, we have reached our goal of designing a model with the ability to predict values as accurately as possible with given dataset. The best model for the task is Random Forest

We believe that the developed model can be a valuable tool for policymakers and law enforcement agencies in understanding and predicting crime rates in new regions.