

АРС #20 Задание 1

ПРОГНОЗИРОВАНИЕ УРОВНЯ ПРЕСТУПНОСТИ И АНАЛИЗ ФАКТОРОВ ВЛИЯНИЯ

Абдыкаримов Амир
Даменова Дана

ДАТАСЕТ

● Crimes in US Communities Dataset

- 1) Датасет предоставлен за 2018 год
- 2) Датасет описывает демографические данные и количество преступления в разных сообществах США
- 3) Демографические данные включают в себя: популяцию, возраст, расу, доход, образование, бедность, иммиграцию, полицию, структуру семьи.
- 4) Данные о преступлениях включают в себя: вид преступления и количество преступлений в данном сообществе



ПРОПУЩЕННЫЕ ЗНАЧЕНИЯ

```
In [3]: df_crime.isna().sum().sort_values(ascending=False)
```

```
Out[3]: PctPolicBlack      1872
        PolicAveOTWorked  1872
        PolicOperBudg     1872
        LemasPctPolicOnPatr 1872
        LemasGangUnitDeploy 1872
        ...
        PctWorkMomYoungKids    0
        PctWorkMom             0
        NumKidsBornNeverMar    0
        PctKidsBornNeverMar    0
        PctHousLess3BR         0
        Length: 146, dtype: int64
```

- Есть колонны, в которых пропущено 84.5% данных
- Для восполнения, используем `mean imputing` для численных значений (вводим среднее значение по штату)
- Удаляем колонны с низкой предиктивной способностью (`communityCode`, `communityName`)
- С полученным датасетом будет проведена дальнейшая работа для всех моделей

```
df_crime.iloc[:, 4:] = df_crime.iloc[:, 4:].apply(pd.to_numeric, errors='coerce')
```

```
transform_numeric_cols = df_crime.iloc[:, 4:].columns
```

```
df_crime[transform_numeric_cols] = df_crime.groupby('state')[transform_numeric_cols].transform(lambda x: x.fillna(x.mean()))
```


ПОДГОТОВКА ДАННЫХ ДЛЯ РЕГРЕССИИ

1 Удаление колонок

Удаляем колонны, содержащие преступления, отличные от таргета (интересующей колонны), чтобы основываться только на демографических и социальных данных

3 Таргет Энкодинг

Классовые колонны по типу штата, округа, не могут быть считаны моделью, потому применяется target encoding – метод, присваивающий классу среднее значение таргета (например, если в штате Вирджиния в среднем 3.5 убийства, “Вирджиния” принимает значение 3.5)

2 Удаление выбросных данных

Удаление аутлаеров внутри таргета используя IQR, чтобы выбросные значения не искажали данные

```
def train_models_for_targets(data, targets):
    results = {}

    cols_to_drop = [
        "murders", "murdPerPop", "rapes", "rapesPerPop", "robberies", "robberPerPop",
        "assaults", "assaultPerPop", "burglaries", "burglPerPop", "larcenies",
        "larcPerPop", "autoTheft", "autoTheftPerPop", "arsons", "arsonsPerPop",
        "ViolentCrimesPerPop", "nonViolPerPop", "communityName"
    ]

    for target in targets:
        print(f"Processing target: {target}")

        X = data.drop(columns=cols_to_drop + [target], errors='ignore')
        y = data[target]

        # Удаление аутлаеров / выбросов
        Q1 = y.quantile(0.25)
        Q3 = y.quantile(0.75)
        IQR = Q3 - Q1
        data_filtered = data[(y >= Q1 - 1.5 * IQR) & (y <= Q3 + 1.5 * IQR)]
        y_filtered = data_filtered[target]
        X = X.loc[data_filtered.index]

        # Таргет энкодинг / преобразование классов в численные значения для модели
        categorical_columns = ["state", "countyCode"]

        # Вычисление средних значений по штатам и округам для преобразования (таргет энкодинга)
        state_target_encoding = data_filtered.groupby('state')[target].mean()
        county_target_encoding = data_filtered.groupby('countyCode')[target].mean()
```

ПОДГОТОВКА ДАННЫХ ДЛЯ РЕГРЕССИИ

4 Выбор приоритетных колонн

Применение random forest selector для определения 20 приоритетных колонн

```
# деление на тренировочные и тестовые данные
X_train, X_test, y_train, y_test = train_test_split(X, y_filtered, test_size=0.2, random_state=42)

# Выбираем приоритетные колонны, топ-20, используя случайный лес / random forest
selector = RandomForestRegressor(random_state=42)
selector.fit(X_train, y_train)
importances = selector.feature_importances_
top_features_idx = np.argsort(importances)[-20:]
top_features = X.columns[top_features_idx]

X_train = X_train[top_features]
X_test = X_test[top_features]

# Сохраним информацию о приоритетных колоннах в csv
feature_weights = pd.DataFrame({"Feature": top_features, "Importance": importances[top_features_idx]})
feature_weights.sort_values(by="Importance", ascending=False, inplace=True)
feature_weights.to_csv(f"feature_weights/{target}.csv", index=False)

# Сохраним также в качестве графика для визуализации на сайте
plt.figure(figsize=(10, 6))
plt.barh(feature_weights["Feature"], feature_weights["Importance"], color="skyblue")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.title(f"Feature Importance for {target}")
plt.savefig(f"images/{target}.jpg")
plt.close()

# Тренировка моделей
models = {
    "RandomForest": RandomForestRegressor(random_state=42),
    "KNN": KNeighborsRegressor(),
    "XGBoost": XGBRegressor(random_state=42),
    "DecisionTree": DecisionTreeRegressor(random_state=42)
}
```

5 Тренировка моделей

Тренировка на 20 выбранных колоннах моделей Linear regression, KNN regressor, Random Forest, XGBoost.

Причина выбора – малое кол-во данных, задача регрессии. Модель с наименьшим RMSE выбирается и экспортируется используя joblib

```
metrics = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Вычисление метрик
    rmse = root_mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    metrics[model_name] = {"RMSE": rmse, "R2": r2, "MAE": mae}
print(X_train.columns)
# Выбираем лучшую модель основываясь на самом низком RMSE
best_model_name = min(metrics, key=lambda m: metrics[m]["RMSE"])
best_model = models[best_model_name]
print(f"Best model for {target}: {best_model_name} with RMSE {metrics[best_model_name]['RMSE']}")

# Сохраняем модель в расширении pkl чтобы подключить к бэкенду
joblib.dump(best_model, f"models/{target}.pkl")

# Добавляем полученные результаты в json файл
results[target] = {
    "BestModel": best_model_name,
    "Mean": y.mean(),
    **metrics[best_model_name]
}

# Сохраняем метрики
results_path = "results.json"
with open(results_path, "w") as f:
    json.dump(results, f, indent=4)
print(f"Results saved to {results_path}")
```

ОТЛИЧИЯ ДЛЯ КЛАССИФИКАЦИИ И РЕГРЕССИИ С ПРЕСТУПЛЕНИЯМИ

Регрессия с преступлениями

Учитываются другие виды преступления при предсказании таргета для повышения точности в случаях, если пользователю известны преступления иного типа (например, известны кражи, но предсказываются убийства)

Классификация

Вместо регрессии предсказывается Высокая/Низкая степень преступлений (1/0). Для этого мы использовали нормализацию данных по колоннам преступлений (чтобы придать равный вес всем преступлениям) и сложили их. Топ-30% по полученным показателям были присвоены High crime (1), остальным Low crime (0). Исходя из этого была натренирована модель.

```
x = data.drop(columns= [target,
"murdPerPop",
"rapesPerPop",
"robberPerPop",
"assaultPerPop",
"burglPerPop",
"larcPerPop",
"autoTheftPerPop",
"arsonsPerPop",
"ViolentCrimesPerPop",
"nonViolPerPop", 'communityName'], errors='ignore')
```

```
In [3]: import pandas as pd
from sklearn.preprocessing import MinMaxScaler

crime_columns = [
    "murders", "rapes", "robberies", "assaults",
    "burglaries", "larcenies", "autoTheft", "arsons"
]

scaler = MinMaxScaler()
df_crimes[crime_columns] = scaler.fit_transform(df_crimes[crime_columns])

df_crimes["Normalized Crime Score"] = df_crimes[crime_columns].sum(axis=1)

threshold = df_crimes["Normalized Crime Score"].quantile(0.7)

def classify_crime(score):
    if score < threshold:
        return 0
    else:
        return 1

df_crimes["High_Crime_Level"] = df_crimes["Normalized Crime Score"].apply(classify_crime)
```

РЕЗУЛЬТАТЫ

	BestModel	Mean	RMSE	R2	MAE
murders	RandomForest	7.764786	1.082369	0.522194	0.755831
rapes	RandomForest	29.491831	7.16491	0.69249	4.493944
robberies	RandomForest	237.915598	19.124502	0.748135	12.286311
assaults	RandomForest	326.027948	58.522409	0.57234	38.816215
burglaries	RandomForest	761.645189	118.8001	0.783119	78.755725
larcenies	RandomForest	2138.107175	369.33812	0.754332	267.231861
autoTheft	RandomForest	516.405406	64.978221	0.675757	42.402884
arsons	RandomForest	32.832695	6.369862	0.490436	4.615692

Лучшая модель - Random forest для всех задач регрессии с отклонением 5-13% к среднему (MAE/среднее *100%)

Лучшая модель - Random forest для всех задач регрессии с преступностями с отклонением 3.7-13% к среднему (MAE/среднее *100%)

Results from the JSON file:

	BestModel	Mean	RMSE	R2	MAE
murders	RandomForest	7.764786	1.100679	0.505892	0.764936
rapes	RandomForest	29.491831	6.785442	0.7242	4.156959
robberies	RandomForest	237.915598	15.914334	0.825593	9.906997
assaults	RandomForest	326.027948	55.422811	0.616442	34.878019
burglaries	RandomForest	761.645189	93.969862	0.864305	60.734057
larcenies	RandomForest	2138.107175	333.45514	0.799749	230.372267
autoTheft	RandomForest	516.405406	54.631456	0.770797	35.390071
arsons	RandomForest	32.832695	6.407274	0.484433	4.578442

Model Comparison Results:

	Model	Accuracy	Precision	Recall	F1-Score
3	XGBoost	0.918736	0.918586	0.918736	0.917589
2	Random Forest	0.911964	0.912444	0.911964	0.910210
0	Logistic Regression	0.869074	0.869815	0.869074	0.864590
1	KNN Classifier	0.860045	0.862150	0.860045	0.854022

92% Точность для классификации
Высокой/Низкой степени преступности
Лучшая модель - XGBoost