**Введение**

**Цель проекта**

Целью данного проекта является применение алгоритма XGBoost для решения сложной задачи предсказательного анализа. XGBoost, или eXtreme Gradient Boosting, - это передовой алгоритм машинного обучения, известный своей высокой производительностью и гибкостью. Он широко используется в различных областях для решения задач классификации и регрессии.

**Особенности XGBoost**

XGBoost является одним из самых популярных и эффективных алгоритмов машинного обучения на сегодняшний день. Он основан на принципах градиентного бустинга и отличается улучшенной производительностью и способностью эффективно работать с большими объемами данных. Алгоритм также поддерживает регуляризацию, что помогает предотвратить переобучение модели.

**Библиотеки и инструменты**

В рамках проекта используются следующие основные библиотеки и инструменты анализа данных:

**Pandas:** Эта библиотека предназначена для обработки и анализа структурированных данных. Она обеспечивает удобные средства для манипуляции с наборами данных и их очистки.

**NumPy:** Библиотека, используемая для выполнения вычислений с массивами и матрицами. Она является основой для многих других библиотек анализа данных.

**XGBoost:** Центральный инструмент проекта, представляющий собой мощную библиотеку для градиентного бустинга.

**Основы XGBoost**

**XGBoost** (eXtreme Gradient Boosting) - это оптимизированный алгоритм градиентного бустинга. Он характеризуется ускоренным обучением, эффективностью в использовании ресурсов и поддержкой различных функций потерь. Алгоритм особенно эффективен для работы с большими и сложными наборами данных и предоставляет гибкие возможности для настройки гиперпараметров.

**Описание данных**

**Исходные данные**

Для проекта использовался специфический набор данных, содержащий различные характеристики и параметры, необходимые для решения задачи с помощью XGBoost.

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 354369 entries, 0 to 354368

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 DateCrawled 354369 non-null object

1 Price 354369 non-null int64

2 VehicleType 316879 non-null object

3 RegistrationYear 354369 non-null int64

4 Gearbox 334536 non-null object

5 Power 354369 non-null int64

6 Model 334664 non-null object

7 Kilometer 354369 non-null int64

8 RegistrationMonth 354369 non-null int64

9 FuelType 321474 non-null object

10 Brand 354369 non-null object

11 Repaired 283215 non-null object

12 DateCreated 354369 non-null object

13 NumberOfPictures 354369 non-null int64

14 PostalCode 354369 non-null int64

15 LastSeen 354369 non-null object

dtypes: int64(7), object(9)

memory usage: 43.3+ MB

**Предобработка данных**

Перед обучением модели данные прошли процесс предобработки. Этот процесс включал очистку данных от пропусков и аномалий, а также преобразование категориальных переменных в числовой формат. Особое внимание было уделено корректному форматированию данных, чтобы обеспечить максимальную эффективность алгоритма XGBoost.

total = df.isnull().sum().sort\_values(ascending = False)

percent = round(df.isnull().sum().sort\_values(ascending = False)/len(df)\*100,2)

pd.concat([total, percent], axis=1, keys=['Total','Percent']).style.background\_gradient(cmap='Greens')

df.head()

df\_for\_drop = np.round(100 - (len(period) \* 100 / len(df)), 3)

print('% машин под удаление', df\_for\_drop)

not\_rem = df.loc[df['repaired'].isna()]['registrationyear'].value\_counts()

not\_rem\_perc = not\_rem.head(10).sum() / df.loc[df['repaired'].isna(), 'price'].count()

print(not\_rem.head(10))

print()

print('Автомобили без указания ремонта, составляют {:.2%} от общего количества'.format((not\_rem\_perc)))

df['notrepaired'] = df['repaired'].fillna(value='no')

df['notrepaired'].head()

df['GearboxUnsorted'] = df['Gearbox'].fillna(value='unsorted')

df['Gearbox'] = df['Gearbox'].fillna('manual')

temp = pd.DataFrame(df.groupby(['Brand'])['Model'].count()).reset\_index()

temp.sort\_values(by='Model', ascending=False).tail()

mod\_dict = df.groupby(['Brand'])['Model'].apply(lambda x: x.mode()[0])

df['Model'] = df['Model'].fillna(df['Brand'].map(mod\_dict))

vt\_dict = dict(df.groupby(['Model'])['VehicleType'].apply(lambda x: x.mode()[0]))

df['VehicleType'] = df['VehicleType'].fillna(df['Model'].map(vt\_dict))

fuel\_dict = df.groupby(['Model'])['FuelType'].apply(lambda x: x.mode()[0])

np.unique(fuel\_dict.values)

df = df.drop\_duplicates()

**Подготовка данных**

**Обработка и очистка данных**

Важным этапом в подготовке данных является их тщательная очистка и обработка. Это включает обработку пропущенных значений, удаление или преобразование аномальных данных и стандартизацию форматов. В проекте особое внимание уделялось анализу и обработке пропусков в данных, чтобы обеспечить точность и надежность модели XGBoost.

**Выбор и инженерия признаков**

Выбор релевантных признаков и их инженерия - ключевые аспекты в подготовке данных для обучения модели. В процессе анализа данных проводился отбор значимых признаков, которые могут оказывать наибольшее влияние на результаты модели. Также проводилась инженерия признаков, включая создание новых признаков на основе существующих данных, что позволило улучшить производительность модели.

**Построение модели**

**Конфигурация модели XGBoost**

Построение модели начинается с определения конфигурации алгоритма XGBoost. Это включает в себя настройку гиперпараметров, таких как скорость обучения, глубина деревьев, количество деревьев и параметры регуляризации. Выбор этих параметров имеет значительное влияние на производительность и обобщающую способность модели.

%%time

dummy = DummyRegressor(strategy='mean')

dummy.fit(x\_train, y\_train)

y\_pred = dummy.predict(x\_valid)

rmse\_dr = np.around(np.sqrt(mean\_squared\_error(y\_valid, y\_pred)),5)

print('RMSE дамми регрессора со средним прогнозом цен = {}'.format(rmse\_dr))

print()

features\_ = pd.get\_dummies(features, drop\_first=True)

target\_ = df['price']

x\_train\_val, x\_test\_, y\_train\_val, y\_test\_ = train\_test\_split(features\_, target\_, test\_size=0.2, random\_state=12345)

x\_train\_, x\_valid\_, y\_train\_, y\_valid\_ = train\_test\_split(x\_train\_val, y\_train\_val, test\_size=0.25, random\_state=12345)

print(x\_train\_.shape[0], x\_valid\_.shape[0], x\_test\_.shape[0])

**Процесс обучения**

Модель обучается на подготовленных данных. В этом процессе данные разделяются на обучающую и тестовую выборки, что позволяет оценить качество модели на независимых данных. Применяется кросс-валидация для проверки стабильности и надежности модели на разных наборах данных.

dtrain = xgb.DMatrix(x\_train\_, label=y\_train\_)

dval = xgb.DMatrix(x\_valid\_)

dtest = xgb.DMatrix(x\_test\_)

%%time

param = {

'max\_depth': 3,

'eta': 1,

'sampling\_method': 'uniform',

'verbosity': 1,

'gamma': 10,

'seed': 12345

}

num\_round = 30

bst = xgb.train(param, dtrain, num\_round)

**Оптимизация модели**

В процессе обучения проводится тонкая настройка гиперпараметров с целью достижения оптимального баланса между точностью предсказаний и избеганием переобучения. Это может включать методы, такие как поиск по сетке или случайный поиск, для систематического исследования пространства гиперпараметров.

%%time

param = {'max\_depth':6,

'eta':0.5,

'min\_child\_weight':4,

'sampling\_method':'uniform',

'verbosity':1,

'seed':12345}

num\_round = 35

bst = xgb.train(param, dtrain, num\_round)

**Результаты и анализ**

**Оценка производительности модели**

После завершения процесса обучения модели XGBoost проводится анализ её производительности. Используются различные метрики, такие как точность, полнота, F-мера и AUC-ROC, для оценки качества модели на тестовых данных. Эти метрики помогают понять, насколько хорошо модель способна различать классы и делать точные предсказания.

**Анализ важности признаков**

Важной частью анализа является оценка влияния различных признаков на предсказания модели. XGBoost предоставляет инструменты для оценки важности признаков, что позволяет выявить наиболее значимые переменные и понять, как они влияют на результаты классификации.

**Выводы и рекомендации**

На основе результатов анализа формулируются выводы относительно качества и эффективности модели. Обсуждаются возможные направления для улучшения модели, включая доработку признаков, дальнейшую оптимизацию гиперпараметров и использование дополнительных методов для улучшения обобщающей способности модели.

**Заключение**

**Итоги работы с XGBoost**

В ходе проекта была успешно реализована и оценена модель XGBoost, которая показала хорошие результаты в решении поставленной задачи предсказательного анализа. Благодаря глубокому анализу данных и тщательной настройке гиперпараметров, модель демонстрирует высокую точность и способность к обобщению.

**Вклад модели в решение задачи**

Модель XGBoost оказалась эффективным инструментом, способным обрабатывать сложные наборы данных и выявлять важные закономерности. Её гибкость и мощь делают её ценным ресурсом для широкого спектра задач анализа данных.

**Перспективы дальнейшего развития**

На основе полученных результатов можно рекомендовать дальнейшие направления исследования, включая интеграцию дополнительных данных, эксперименты с различными конфигурациями модели и применение ансамблевых методов для дальнейшего улучшения точности и надежности предсказаний.