Харківський національний університет імені В.Н. Каразіна
Факультет математики і інформатики
Кафедра прикладної математики

Кваліфікаційна робота магістра
на тему
«Інтегрований аналіз та прогнозування цін автострахування на основі алгоритмів машинного навчання та статистичного аналізу»

Виконав:
студент групи МП-62
2 курсу
спеціальність 113 – прикладна математика
освітньо-професійна програма «Прикладна математика»
Резуненко Сергій Олександрович
Науковий керівник: к.ф.-м.н., доцент
Стєпанова Катерина Вадимівна
Рецензент: к.т.н., доцент Базілевич Ксенія Олексіївна

Харків 2023 рік

Abstract

In the current work, data analysis and preparation, model training, study of work mechanisms and comparison of the results of different machine learning algorithms and gradient boosting mechanisms for predicting auto insurance prices and analyzing its dependencies have been carried out.

The work includes an overview of gradient boosting methods in the problems of predicting auto insurance prices. A comparison of the efficiency of gradient boosting models XGBoost, LightGBM and CatBoost was carried out. Geodata collection and processing are performed to determine the impact of geographic location on auto insurance prices. Key factors affecting insurance prices have been identified.

The practical application of the developed methods is due to the training of models and the analysis of their effectiveness. Analysis and determination of the importance of various features in predicting auto insurance prices is carried out using SHAP, Permutation Importance and other methods.

Анотація

У даній роботі проведено аналіз та підготовку даних, тренування моделі, вивчення механізмів роботи та порівняння результатів різних алгоритмів машинного навчання та механізмів градієнтного бустингу задля прогнозування цін на автострахування та аналізу її залежностей.

У роботі присутній огляд методів градієнтного бустингу в задачах прогнозування цін на автострахування. Проведено порівняння ефективності моделей градієнтного бустингу XGBoost, LightGBM та CatBoost. Для визначення впливу географічного положення на ціни автострахування виконується збір та обробка геоданих. Здійснена ідентифікація ключових факторів, що впливають на ціни страховки.

Практичне застосування розроблених методів обумовлено тренуванням моделей і аналізом їх результативності. Аналіз та визначення важливості різних ознак у прогнозуванні цін на автострахування здійснюється за допомогою методів SHAP, Permutation Importance та інших.

Зміст

Вступ

Сучасна галузь автострахування стала ключовою складовою фінансового ринку, що зумовлено зростанням кількості автотранспорту та прискореним розвитком технологій. Цей розвиток породжує потребу в інноваційних методах аналізу та прогнозування цін на автострахування. Традиційні підходи до визначення страхових премій можуть виявитися недостатньо ефективними в умовах зростаючої складності ризиків.

Актуальність та значимість проблеми: У контексті цих викликів, дипломна робота магістра присвячена розробці та впровадженню інтегрованого аналізу цін на автострахування. Цей підхід поєднує алгоритми машинного навчання та статистичний аналіз для точного та передбачуваного визначення страхових премій. Такий інтегрований підхід має потенціал надавати більш точні прогнози порівняно з традиційними методами, що робить його важливим напрямком досліджень.

Окрім цього, автоматизація відіграє ключову роль у виконанні складних обчислень та обробці великого обсягу даних. В контексті дослідження цін на автострахування, використання автоматизованих інструментів для збору, обробки та аналізу даних може значно полегшити роботу та забезпечити ефективність процесу. Цей підхід до автоматизації сприяє не лише точності результатів, але й робить аналітичні завдання більш масштабованими та адаптивними до змін у ринкових умовах.

Отже, ця дипломна робота спрямована на розробку та валідацію інтегрованого аналізу цін на автострахування, базованого на алгоритмах машинного навчання та статистичному аналізі, з акцентом на автоматизації процесів. Це дозволить забезпечити страхові компанії необхідними інструментами для точного прогнозування та ефективного управління ризиками в динамічному середовищі ринку автострахування.

Розділ 1. Градієнтний бустинг

Методи градієнтного бустингу [1-7], зокрема такі бібліотеки як XGBoost, LightGBM, та CatBoost, широко використовуються в задачах прогнозування цін на автострахування через їхню ефективність та здатність працювати з різноманітними видами даних. Серед аспектів їхнього застосування можна виділити:

Обробка великих обсягів даних стає важливою у завданнях передбачення вартості автострахування, і методи градієнтного бустингу надають ефективні засоби для цього. Це стає особливо критичним у випадках, де маємо значні обсяги історичних даних та різноманітних параметрів.

Керування різноманітністю даних є ще однією важливою перевагою градієнтного бустингу. Метод дозволяє ефективно враховувати різноманіття та взаємодію різних факторів, які впливають на вартість автострахування. Це особливо корисно, оскільки ціни можуть залежати від різних чинників, таких як історія водія, технічний стан автомобіля, регіональні особливості тощо.

Автоматичний відбір ознак є ще однією вагомою перевагою методів градієнтного бустингу. Ці методи можуть автоматично визначати важливість різних ознак при прогнозуванні цін, що дозволяє ефективно використовувати модель для визначення ключових факторів, впливаючи на вартість страхових премій.

Регулювання гіперпараметрів, яке зазвичай включено в бібліотеки градієнтного бустингу, надає широкі можливості для оптимізації моделі під конкретну задачу та набір даних.

Наприкінці, важливо відзначити, що деякі бібліотеки градієнтного бустингу, такі як CatBoost, ефективно взаємодіють з категоріальними ознаками, що стає важливим у випадку включення таких ознак у модель. Усі ці аспекти допомагають покращити точність прогнозувань вартості автострахування та враховувати складність даної задачі. Опис математичної моделі:

Припустимо, що є функція втрат

$$L(y, F(x)),$$

де: y - цільова змінна (спостережуване значення), $F(x)$ - поточне прогнозоване значення на вхідних даних x.

Мета полягає в тому, щоб навчити послідовність моделей $F_m(x)$ так, щоб мінімізувати середню функцію втрат на всіх навчальних прикладах:

$$\sum L(y, F(x)) \to min$$

Градієнтний бустинг використовує градієнт цієї функції втрат по відношенню до поточної моделі $F(x)$ для навчання наступної моделі. Ми оновлюємо поточну модель, додаючи градієнт функції втрат помножений на невеликий коефіцієнт (швидкість навчання або learning rate). Це допомагає покращити прогнози там, де поточна модель робить помилки.

На кожній ітерації ми навчаємо нову модель $F_m(x)$, додаючи градієнт функції втраті по відношенню до поточної апроксимації $F_{m-1}(x)$. Тобто:

$$F_m(x) = F_{\{m-1\}}(x) + \eta \nabla L(y, F_{\{m-1\}}(x))$$

де $F_m(x)$ - нова модель на m-й ітерації,

$F_{\{m-1\}}(x)$ - попередня модель на $(m-1)$-й ітерації,

η - швидкість навчання (learning rate),

$\nabla L(y, F_{\{m-1\}}(x))$ - градієнт функції втрат по відношенню до попередньої моделі.

Цей процес повторюється протягом кількох ітерацій до того моменту, поки не буде досягнута достатня точність. На кожній ітерації вводиться нова модель, яка виправляє попередні помилки, сприяючи підвищенню точності прогнозів у моделі градієнтного бустингу.

Розділ 2. Підготовка даних

2.1 Аналіз та підготовка даних

У якості вхідних даних для аналізу беруться дані про ціни на автострахування на території UK(Об’єднане Королівство) разом зі набором супутніх ознак. Оригінальний набір даних нараховує 1160375 рядків(спостережень) та такі ознаки, як: Quote Reference, Quote Date, Vehicle Group 50, Vehicle Estimated Value, Date of Birth, Postcode, Voluntary Excess, Type of driving license, Vehicle Use, UK resident from, Number of drivers, Job Title, No Claim Bonus, Policy Price, Median Price, Avg top 5. Також на певному етапі обробки даних, досліджуваний набір даних поєднується з географічними даними. Ознайомившись з методологію та роботами на тему приведення даних до робочого виду [8-9], будо реалізовано наступні операції на вхідними даними.

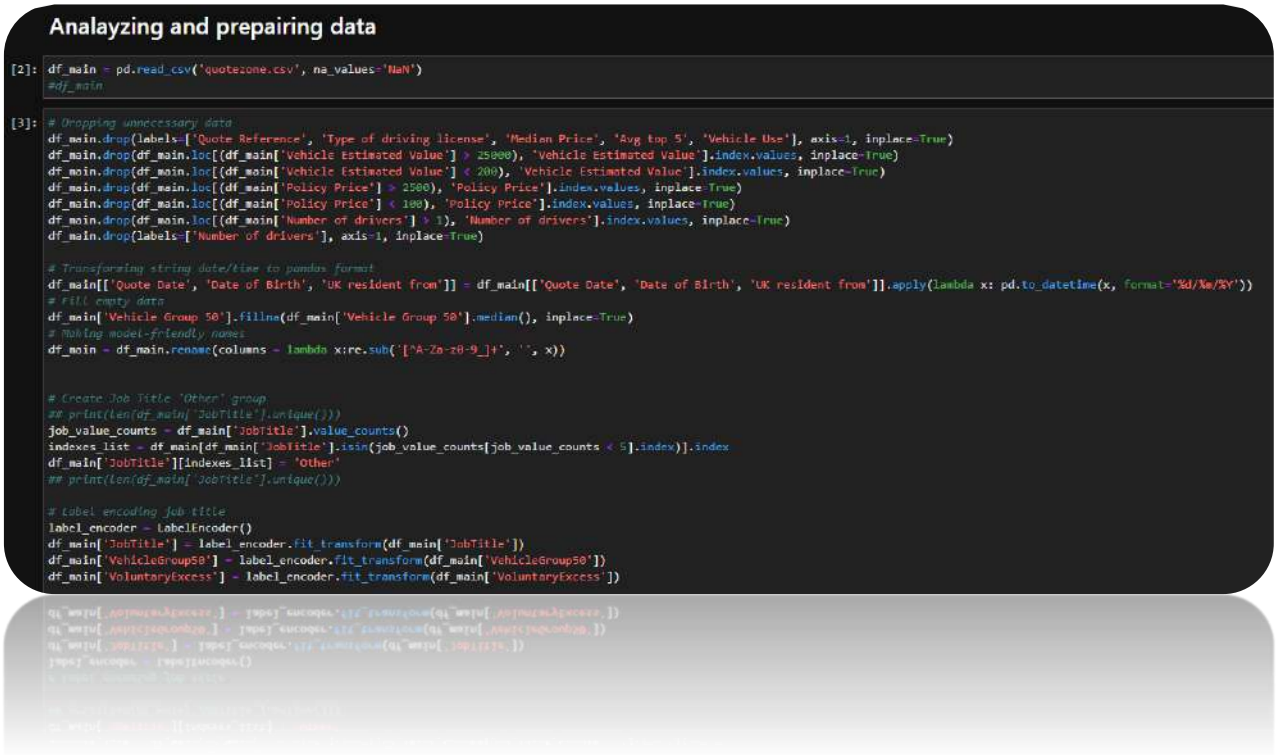


Рисунок 2.1. Перший етап підготовки даних

Початком обробки даних стає видалення декількох стовпців, що не беруть участь у нашому аналізі та введення обмежень для деяких стовпців,

таких як, 'Vehicle Estimated Value', 'Policy Price', та 'Number of drivers', щоб вони відповідали певним критеріям. (рис.2.1).

Слідом йде трансформація дат та заповнення пропущених даних. Рядки дат перетворюються в об'єкти типу дати у Pandas, порожні значення 'Vehicle Group 50' заповнюються значенням медіани. Стовпці перейменовуються, стаючи більш придатними для моделювання. Групування рідкісних значень в стовпці 'JobTitle' під загальною категорією 'Other'.

Label Encoding застосовується до категоріальних ознак для їхнього перетворення в числові.

```
[4]: df_matched = pd.read_csv('matched.csv', na_values='NaN')
     df_unmatched = pd.read_csv('unmatched.csv', na_values='NaN')

[5]: df_matched.set_index('input_row', inplace=True)
     df_matched.index.name = None
     df_matched.index -= 1
     df_matched['IMDScore'] = df_matched.EIMD_2015_SCORE1115jul
     df_matched['IMDDecile'] = df_matched.EIMD_2015_DECILE1115jul
     df_matched.drop(labels=['EIMD_2015_SCORE1115jul', 'EIMD_2015_DECILE1115jul', 'standard_postcode'], axis=1, inplace=True)
     df_matched.drop(df_matched.columns[0], axis=1, inplace=True)
     df_matched['IMDScore'].fillna(99, inplace=True)
     df_matched['IMDDecile'].fillna(0, inplace=True)

[6]: df_unmatched.set_index('input_row', inplace=True)
     df_unmatched.index.name = None
     df_unmatched.index -= 1
     df_unmatched['IMDScore'] = 99
     df_unmatched['IMDDecile'] = 0
     df_unmatched.drop(df_unmatched.columns[0], axis=1, inplace=True)

[7]: df_mat_unmat = pd.concat([df_matched, df_unmatched])
     df_mat_unmat.sort_index(inplace=True)
     df_mat_unmat.index = df_main.index

[8]: df_main = pd.concat([df_main, df_mat_unmat], axis=1)
     df_main
```

Рисунок 2.2. Об’єднання даних з додатковими з файлів 'matched.csv' та 'unmatched.csv'

Шляхом, продемонстрованим на (рис. 2.2), дані набувають відповідного вигляду для подальшого їх використання в моделі прогнозування цін на автострахування, враховуючи важливі аспекти, такі як обробка пропущених значень, видалення зайвих даних та трансформація категоріальних ознак.

Додавання географічних ознак за поштовими індексами відбувається за рахунок завантаження з файлу 'postcodes.csv', який містить географічні координати (широту та довготу) для поштових індексів. Це може бути корисно для моделювання залежностей від розташування.

Додавання області та обробка категорій для поштових індексів. Префікси поштових індексів використовуються задля визначення областей та встановлення категорій. Невизначені області заповнюються значенням 200.

Working with posctodes

```
9]: postcode_features = pd.read_csv('quotezone/retrain/postcodes.csv', index_col='Postcode',
                                    usecols=['Postcode', 'Latitude', 'Longitude'])
    # Add area
    prefixes = df_main['Postcode'].str.split(' ', n=1, expand=True)[0].str.upper()
    areas_ratings_df = pd.read_csv('quotezone/retrain/Postcode.Ratings.csv')
    area_mapping = dict(zip(areas_ratings_df['Postcode.Prefix'], areas_ratings_df['Postcode.Mapping']))
    df_main['Area'] = prefixes.map(area_mapping)
    df_main['Area'] = df_main['Area'].fillna(200).astype(int)
    # Add latitude and longtitude of area
    df_main['Latitude'] = df_main.Postcode.map(postcode_features['Latitude'])
    df_main['Longitude'] = df_main.Postcode.map(postcode_features['Longitude'])

    # Create postcode 'Other' group
    ## print(len(df_main['Postcode'].unique()))
    posctodes_counts = df_main['Postcode'].value_counts()
    indexes_list = df_main[df_main['Postcode'].isin(posctodes_counts[posctodes_counts < 10].index)].index
    df_main.Postcode[indexes_list] = 'Other'
    ## print(len(df_main['Postcode'].unique()))

    # Label encoding postcodes
    df_main.Postcode = label_encoder.fit_transform(df_main.Postcode)
    df_main.IMDScore = label_encoder.fit_transform(df_main.IMDScore)
    df_main.IMDDecile = label_encoder.fit_transform(df_main.IMDDecile)

    df_main
```

Рисунок 2.3. Додавання географічних координат та обробка категорій для поштових індексів та Label Encoding

Додаються географічні координати (широта та довгота на основі поштових індексів. Створення групи 'Other' для рідкісних значень поштових індексів та Label Encoding. Значення поштових індексів групуються під

загальною категорією 'Other' та застосовується Label Encoding до категорійних ознак, таких як 'Postcode', 'IMDScore' та 'IMDDecile' (рис. 2.3).

На цьому етапі до набору даних додавалися географічні та категоріальні ознаки, щоб розширити інформаційні можливості для моделі прогнозування цін на автострахування.

Replacing dates with ages

```
df_main['AgeOfVehicle'] = 0
df_main.loc[(df_main['QuoteDate'] < '01/07/2019'), 'AgeOfVehicle'] = 2019 - df_main.loc[(df_main['QuoteDate'] < '01/07/2019'), 'VehicleYearofManufacture']
df_main.loc[(df_main['QuoteDate'] > '01/07/2019'), 'AgeOfVehicle'] = 2020 - df_main.loc[(df_main['QuoteDate'] > '01/07/2019'), 'VehicleYearofManufacture']

df_main['Age'] = np.around((df_main['QuoteDate'] - df_main['DateofBirth']) / np.timedelta64(1,'Y')).astype(int)
df_main['YearsBeingResidentOfUK'] = np.around( (df_main['QuoteDate'] - df_main['UKresidentfrom']) / np.timedelta64(1,'Y') ).astype(int)

df_main.drop(labels=['DateofBirth', 'UKresidentfrom', 'VehicleYearofManufacture'], axis=1, inplace=True)

df_main['ResidentFromBirth'] = 0
df_main.loc[(df_main['Age'] - df_main['YearsBeingResidentOfUK'] == 0), 'ResidentFromBirth'] = 1
```

Рисунок 2.4. Розрахунок додаткових характеристик

Розрахунок віку автомобіля та інших характеристик відбувається на момент отримання квоти на страхування, використовуючи інформацію про рік виробництва. Розрахунок віку та тривалості проживання в Об'єднаному Королівстві. Стовпці, які більше не потрібні для моделювання, відкидаються (рис. 2.4).

Створення нової бінарної ознаки "ResidentFromBirth", яка позначає, чи особа проживає в Об'єднаному Королівстві з моменту народження.

Splitting data

```
df_train = df_main.loc[(df_main['QuoteDate'] < '2019-07-01')]
df_val = df_main.loc[(df_main['QuoteDate'] >= '2019-07-01') & (df_main['QuoteDate'] < '2019-9-27')]
df_test = df_main.loc[(df_main['QuoteDate'] >= '2019-9-27')]

df_train.drop(labels=['QuoteDate'], inplace=True, axis=1)
df_val.drop(labels=['QuoteDate'], inplace=True, axis=1)
df_test.drop(labels=['QuoteDate'], inplace=True, axis=1)

y_train = df_train['PolicyPrice'].to_numpy()
x_train = df_train.drop(labels=['PolicyPrice'], axis = 1)

y_val = df_val['PolicyPrice'].to_numpy()
x_val = df_val.drop(labels=['PolicyPrice'], axis = 1)

y_test = df_test['PolicyPrice'].to_numpy()
x_test = df_test.drop(labels=['PolicyPrice'], axis = 1)
```

Рисунок 2.5. Розбиття даних

Вищенаведена частина коду (рис. 2.5) розбиває дані на тренувальний, валідаційний та тестовий набори, видаляє стовпець 'QuoteDate' та формує вектори міток (y) та матриці ознак (X) для тренування, валідації та тестування.

Дані розділяються на три частини: тренувальний (до 1 липня 2019 року), валідаційний (від 1 липня 2019 року до 27 вересня 2019 року) та тестовий (після 27 вересня 2019 року) набори.

Стовпець 'QuoteDate' видаляється з усіх трьох частин даних, оскільки він більше не використовується для моделювання.

Формування векторів міток (y) та матриць ознак (X). Для кожного набору даних (тренувальний, валідаційний, тестовий) формуються вектори міток (ціни полісу) та матриці ознак (всі інші ознаки).

2.2 Підготовка моделі

Grid Search Function

```
#split_index = [-1 if x in x_train.index else 0 for x in x_main.index]

# Use the list to create PredefinedSplit
# pds = PredefinedSplit(test_fold = split_index)
# print(split_index[600000:600030])

cat_features = ['VoluntaryExcess',
                'Postcode',
                'JobTitle',
                'Area',
                'VehicleGroup50',
                'ResidentFromBirth',
                'IMDScore', 'IMDDecile']

def grid_cv(model, param_grid):
    grid_search = GridSearchCV(model, param_grid, cv=10, scoring='accuracy', n_jobs=-1)
    t_start = time.time()
    # grid_search.fit(train_examples, train_labels)
    grid_search.fit(x_main, y_main)
    t_end = time.time()
    print('Model {} best accuracy score is {}'.format(model.__class__.__name__, grid_search.best_score_))
    print('Time for training is {} seconds'.format(t_end - t_start))
    print('Best parameters:{}'.format(grid_search.best_params_))
    return grid_search.best_estimator_
```

Рисунок 2.6. Кат. змінні та функція $grid\_cv$

Створюється список категоріальних ознак, які будуть використовуватися при налаштуванні моделі. Визначається функція $grid\_cv$, яка приймає модель та сітку параметрів для решітчастого пошуку. Функція виконує решітчастий пошук на заданих даних (x_main, y_main), оцінює точність та виводить найкращі параметри та модель.

Виклик функції для кожної моделі. Визначається модель (у цьому випадку RandomForestClassifier) та сітку параметрів для решітчастого пошуку. Викликається функція $grid\_cv$ для налаштування гіперпараметрів та отримання найкращої моделі. Цей код є основою для налаштування гіперпараметрів моделі на основі категоріальних ознак (рис 2.6).

Розділ 3. Огляд моделей градієнтного бустингу XGBoost, LightGBM та CatBoost. Огляд SHAP

3.1.1 XGB Regressor

```
xgb_params = {
    'learning_rate': 0.001,
    'max_depth': 10,
    'n_estimators': 300
}

xgbr = xgb.XGBRegressor(n_jobs=-1, verbosity=3, **xgb_params)
xgbr.fit(x_train, y_train, early_stopping_rounds=5, verbose=True, eval_set=[(x_val, y_val)])
pred_xg = xgbr.predict(x_test)
```

Рисунок 3.1. XGBoost

На рис. 3.1 визначаються параметри для моделі XGBoost, такі як швидкість навчання (learning_rate), глибина дерева (max_depth) та кількість дерев у ансамблі (n_estimators), а також створюється екземпляр XGBoost регресора (xgb.XGBRegressor) з використанням визначених параметрів та навчається на тренувальних даних (x_train, y_train). Параметри, такі як early_stopping_rounds та eval_set, використовуються для зупинки навчання, якщо результат на валідаційному наборі не покращується протягом заданої кількості ітерацій.

Прогнозування на тестовому наборі та отримання прогнозів досягається шляхом виклику методу predict для отримання прогнозів на тестовому наборі (x_test). Отримані прогнози зберігаються у змінній pred_xg.

3.1.2 XGBoost (math)

XGBoost (eXtreme Gradient Boosting) - це потужний алгоритм машинного навчання, який використовує градієнтний бустинг для рішення задач регресії та класифікації. Цей метод базується на ідеї об'єднання слабких моделей (зазвичай дерев рішень) в сильний ансамбль.

XGBoost - одна з найпопулярніших і найефективніших реалізацій алгоритму Gradient Boosted Trees, методу навчання під наглядом, який базується на апроксимації функції шляхом оптимізації певних функцій втрат, а також застосування кількох методів регулярізації. Нижче розглядаються основні математичні підґрунтування методу[10-16]:

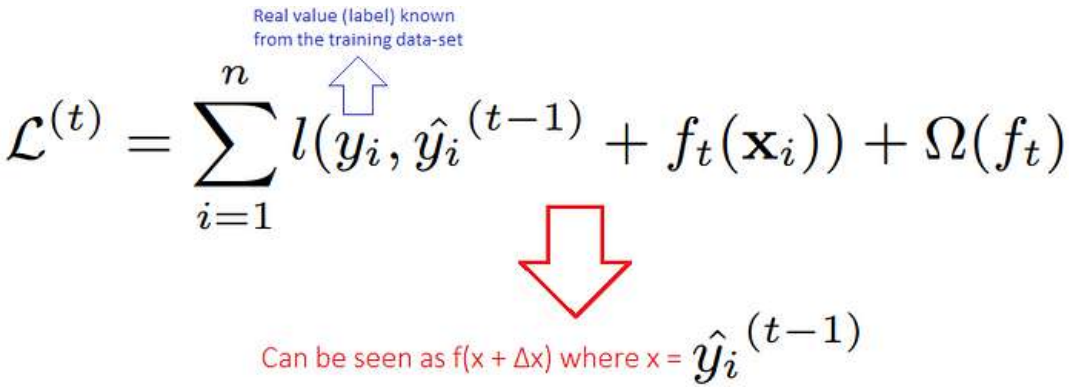
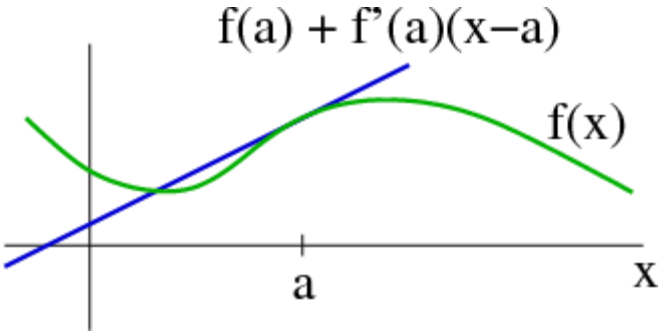


Рисунок 3.2. Цільова функція втрат (t-ітерація)

Очевидна ціль - мінімізація функції втрат.

Задаючись ціллю перетворити вихідну цільову функцію на функцію в евклідовій області, що надає змогу звертатися до традиційних методів оптимізації, використовується наближення Тейлора[17].



$$f(x) \approx f(a) + f'(a)(x-a)$$

$$\Delta x = f_t(\mathbf{x}_i)$$

Рисунок 3.3. Лінійне наближення для функції $f(x)$у точці a

У нашому випадку $f(x)$ є функцією втрат, поки a є прогнозоване значення попереднього кроку $(t-1)$, а Δx – це новий учень, що додається на кроці t. Це справедливо для кожної ітерації t. Цільова функція записується у вигляді функції доданого учня. Результатом обрання наближення Тейлора 2-го порядку буде:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2,$$

$$L^{(t)} \cong \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t),$$

де $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ і $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ - статистики градієнта першого та другого порядку цільової функції

Далі, з ціллю спрощення мінімізації відкидаються постійні частини та формула набуває вигляду:

$$\tilde{L}^{(t)}(q) \cong \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t).$$

Рисунок 3.4. Спрощена функція втрат

Створення наступного учня

Маємо:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$
$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

,

де $I_j = \{i|q(x_i) = j\}$ - набір індексів точок даних j-го листка

Для фіксованої деревоподібної структури q(x) ми можемо визначити оптимальну вагу w^*_j листа j як формули: $w^*_j = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$, тоді формула набуває наступного вигляду:

instances mapped to leaf$_j$

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

Рисунок 3.5. Функція скорингу деревоподібної структури q

Варто зауважити, що наведена вище «функція оцінки якості» повертає мінімальне значення втрат для конкретної деревовидної структури, тобто вихідна функція втрат оцінюється за допомогою оптимальних значень ваги. Отже, для будь-якої даної структури дерева q існує спосіб обчислити оптимальну вагу листя.

Тобто для формування учня потрібно переглянути всі функції та значення для кожної функції та оцінити кожне можливе розділене зменшення втрат:

$$\text{приріст} = \text{втрата}(\text{батьківські екземпляри}) - (\text{втрата (ліва гілка)} +$$

+втрата(права гілка))

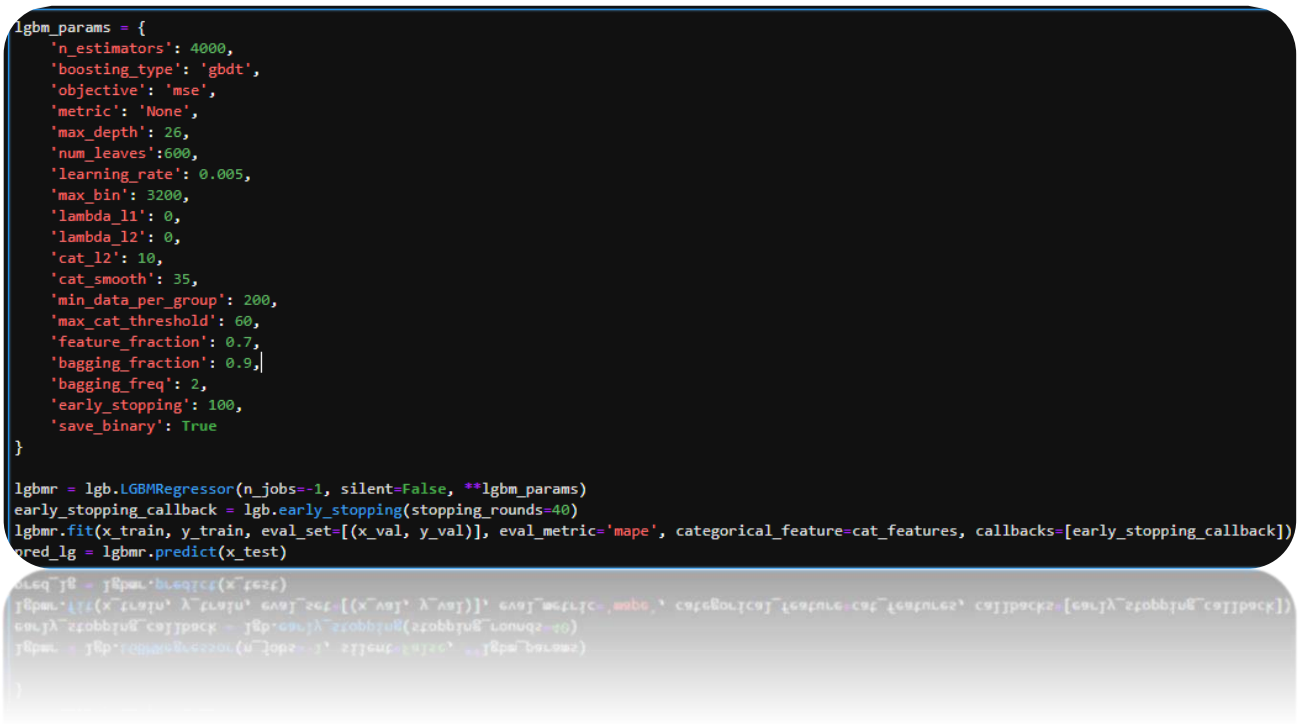
Приріст для найкращого розподілу має бути додатним (і > параметра min_(split_gain)), інакше розвиток гілки припиняється.

Отже, щоб мінімізувати цільову функцію втрат, потрібно знайти її першу та другу похідні (градієнт і Гессе) відносно x.

З практичної точки зору, усі потенційні деревовидні структури q перерахувати неможливо, тому використовується «жадібний» алгоритм, суть якого полягає у додаванні нових листків ітеративно, починаючи з одного листка.

Узагальнюючи модель GBT, яка є сумою CART (дерев), учні намагатимуться мінімізувати цільову функцію втрат, а оцінки на листках, які виконуються роль вагів, мають значення як сума по всіх деревах моделі та піддаються регуляції з ціллю мінімізації втрат.

3.2.1 LightGBM reggressor (code)



```
lgbm_params = {
    'n_estimators': 4000,
    'boosting_type': 'gbdt',
    'objective': 'mse',
    'metric': 'None',
    'max_depth': 26,
    'num_leaves':600,
    'learning_rate': 0.005,
    'max_bin': 3200,
    'lambda_l1': 0,
    'lambda_l2': 0,
    'cat_l2': 10,
    'cat_smooth': 35,
    'min_data_per_group': 200,
    'max_cat_threshold': 60,
    'feature_fraction': 0.7,
    'bagging_fraction': 0.9,
    'bagging_freq': 2,
    'early_stopping': 100,
    'save_binary': True
}

lgbmr = lgb.LGBMRegressor(n_jobs=-1, silent=False, **lgbm_params)
early_stopping_callback = lgb.early_stopping(stopping_rounds=40)
lgbmr.fit(x_train, y_train, eval_set=[(x_val, y_val)], eval_metric='mape', categorical_feature=cat_features, callbacks=[early_stopping_callback])
pred_lg = lgbmr.predict(x_test)
```

Рисунок 3.6. LightGBM

На рис. 3.6 визначаються параметри для моделі LightGBM, такі як кількість дерев (n_estimators), тип алгоритму зростання (boosting_type), функція втрат (objective), метрика втрат (metric), глибина дерева (max_depth), та інші., а також створюється екземпляр LightGBM регресора (lgb.LGBMRegressor) з використанням визначених параметрів, який навчається на тренувальних даних (x_train, y_train). Параметри, такі як eval_set, eval_metric, categorical_feature та callbacks, використовуються для оцінки якості моделі на валідаційному.

Задля отримання прогнозів на тестовому наборі викликається метод predict для отримання прогнозів на тестовому наборі (x_test). Отримані прогнози зберігаються у змінній pred_lg. Після навчання моделі використовуються різні метрики для оцінки її точності.

3.2.2 LightGBM (math)

LightGBM[18-20] — це структура для посилення градієнта, заснована на деревах рішень для підвищення ефективності моделі та зменшення використання пам’яті.

Він використовує дві нові техніки:

- Одностороння вибірка на основі градієнта (GOSS).

Метод GOSS (Gradient-based One-Side Sampling) використовується для підсилення градієнта на тренувальному наборі, який має n екземплярів $\{x_1, \cdots, x_n\}$, де кожен екземпляр x_i представляє собою вектор у просторі X_s з розмірністю s. На кожній ітерації підсилення градієнта від'ємні градієнти функції втрат відносно виходу моделі позначені як $\{g_1, \cdots, g_n\}$. Екземпляри в тренувальному наборі відсортовані в порядку спадання на основі їхніх абсолютних значень градієнта, і екземпляри, які входять до top a × 100% найбільших градієнтів, обираються для створення підмножини A.

Для решти набору Ac, що складається з $(1 - a) \times 100\%$ примірників з меншими градієнтами, випадкова підмножина B розміром $b \times |Ac|$ вибірка. Потім екземпляри розділяють на основі оціненого приросту дисперсії на векторі $V_j\ (d)$ над підмножиною $A \cup B$, де

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right)$$

де

$$Al = x_i \in A : x_{ij} \leq d, \quad Ar = x_i \in A : x_{ij} > d, \quad Bl = x_i \in B : x_{ij} \leq d, \quad Br = x_i \in B : x_{ij} > d$$

Коефіцієнт $\frac{1-a}{b}$ використовується для нормалізації суми градієнтів по B назад до розміру Ac.

- Ексклюзивний пакет функцій (EFB)

Багатовимірні дані, як правило, характеризуються високим рівнем розрідженості, що відкриває можливості для використання підходу майже без

втрат у меті зменшення кількості функцій. Зокрема, у розрідженому просторі ознак багато функцій взаємовиключні, тобто вони ніколи не приймають ненульових значень одночасно. Ексклюзивні функції можна безпечно об’єднати в одну функцію, створюючи так званий набір ексклюзивних функцій. Внаслідок цього складність конструювання гістограми змінюється з O(дані × характеристика) на O(дані × група), при цьому кількість груп менша за кількість функцій. Таким чином, швидкість навчання покращується без втрат точності.

Ці методи відповідають обмеженням алгоритму на основі гістограми, який в основному використовується в усіх структурах GBDT (Gradient Boosting Decision Tree). Дві методики GOSS і EFB, описані вище, формують характеристики алгоритму LightGBM. Вони поєднуються разом, щоб зробити модель ефективнішою та надати їй передову перевагу над іншими фреймворками GBDT.

Переваги моделі LightBGM

- Алгоритм LightBGM вирізняється швидкістю і вищою точністю;
- Має менше споживання пам'яті;
- Підтримує паралельне та розподілене навчання GPU;
- Здатний обробляти великі дані;

3.3.1 CatBoost (code)

```
cat_params = {
    'random_seed': 1,
    'iterations': 2000,
    'depth': 16,
    'learning_rate': 0.02,
    'l2_leaf_reg': 1.1,
    'border_count': 300,
    'loss_function': 'MAE',
    'eval_metric': 'MAPE',
    'early_stopping_rounds': 20,
    #'task_type': 'GPU'
    #'boosting_type': 'Ordered'
    #'max_bin': 3200
}

cb = CatBoostRegressor(**cat_params)
cb.fit(x_train, y_train, eval_set=[(x_val, y_val)], cat_features=cat_features, verbose=20)
```

Рисунок 3.6. CatBoost

Вищенаведений код (рис. 3.6) використовує бібліотеку CatBoost для навчання регресора та прогнозування цін на автострахування на тестовому наборі даних.

Визначаються параметри для моделі CatBoost, такі як кількість ітерацій (iterations), глибина дерева (depth), швидкість навчання (learning_rate), регуляризація L2 (l2_leaf_reg), а також створюється екземпляр CatBoost регресора (CatBoostRegressor) з використанням визначених параметрів та навчається на тренувальних даних (x_train, y_train). Параметри, такі як eval_set, cat_features, verbose та інші, використовуються для оцінки якості моделі на валідаційному наборі та відображення прогресу навчання. Код також включає оцінку якості моделі за допомогою MAPE (Mean Absolute Percentage Error) як функції втрат (loss_function) та метрики оцінки (eval_metric). Останні дві опції дозволяють контролювати та визначати, які метрики використовуються для навчання та оцінки моделі.

3.3.2 CatBoost

CatBoost [21-24] - це потужна бібліотека з відкритим вихідним кодом, спеціалізована на градієнтному бустингу на деревах рішень. Вона забезпечує ефективне вирішення завдань класифікації, регресії та ранжування. CatBoost використовує впорядкований бустинг, випадкові перестановки та оптимізацію на основі градієнтів для досягнення високої ефективності на великих та складних наборах даних з категорійними ознаками. Бібліотека підтримує всі типи ознак - числові, категорійні та текстові, що спрощує процес підготовки даних та забезпечує ефективність на різноманітних типах вхідної інформації. CatBoost вирішує проблему перенавчання за допомогою впорядкованого бустингу. Цей підхід дозволяє уникнути витоку цільової змінної та покращує узагальнення моделі. На відміну від інших алгоритмів, CatBoost будує збалансовані дерева з симетричною структурою. Це сприяє ефективній реалізації на CPU, зменшенню часу передбачення та регуляризації для запобігання перенавчанню. CatBoost є інтерпретованою моделлю, надаючи інструменти для аналізу важливості ознак та графіків рішень. Ці інструменти допомагають зрозуміти поведінку моделі та приймати обґрунтовані рішення. CatBoost відзначається швидким та точним передбаченням, особливо на великих та складних об'ємах даних. Вона перевершує конкурентів, таких як XGBoost і LightGBM, завдяки своїм унікальним функціям та технікам, включає детектор перенавчання, який зупиняє навчання при виявленні ознак перенавчання, поліпшуючи загальні характеристики узагальнення моделі. Основна функціональність CatBoost - обробка категоріальних ознак, що робить її ідеальним рішенням для реальних наборів даних з різноманітними типами вхідних змінних. Бібліотека використовує інженерію ознак, оптимізацію дерев рішень та впорядкований бустинг, створюючи комплексний підхід для підвищення точності.

3.4. SHapley Additive exPlanations

(SHapley Additive exPlanations) [25-30] в машинному навчанні є потужним інструментом для пояснення та інтерпретації моделей машинного навчання. "SHapley Additive exPlanations," скорочується як «SHAP» і базується на ідеї чисел Шеплі з теорії ігор.

Нехай існує характеристична функція v, яка кожній комбінації з множини гравців відповідає число - ефективність даної коаліції гравців, що діє спільно. Shapley value визначається для кожного гравця у комбінації з певною коаліцією гравців – числове значення (очікуване значення) ефективності додавання гравця до коаліції. Позначимо за $\Delta(i, S)$ приріст ефективності від додавання гравця і до коаліції гравців S:

$$\Delta(i, S) = v(S \cup i) - v(s)$$

Нехай всього є N гравців. Розглянемо безліч Π усіх можливих упорядкованих гравців. Позначимо за (гравців перед i в Π) безліч гравців, зо стоять перед гравцем і в упорядкованому Π. Значення Шеплі для такого гравця розраховується наступним чином:

$$\phi(i) = \frac{1}{N!} \sum_{\pi \in \Pi} \Delta(i, (\text{гравці перед } i \text{ в } \Pi))$$

Тому ми вважаємо середній приріст ефективності від додавання i-го гравця в коаліцію гравців, існуючих перед ним, за всіма можливими упорядкуванням гравців (кількість елементів суми рівна $N!$).

Ця формула задається аксіоматично, тобто висувається ряд необхідних властивостей і доводиться, що дане рішення є єдиним, яке задовольняє.

Користуючись тим фактом, що $\Delta\ (i, S)$ не залежить від порядку гравців у S, шляхом об'єднанням доданків, формула набуває наступного вигляду:

$$\phi(i) = \frac{1}{N!} \sum_{S \subseteq \{1,2,\dots N\}/i} \frac{|S|!\,(|N| - |S| - 1)!}{N!}\ \Delta\ (i, S)$$

Ця формула, в свою чергу, є виваженою сумою по всіх підмножинах гравців, що не містять гравця i, в якій ваги набувають найбільших значень при $|S| = 0$ чи $|S| = |N|$ та найменші значення при $|S| = \frac{|N|}{2}$.

Shapley regression values надають змогу оцінити внесок кожної ознаки у відповідь моделі f. Зафіксуємо конкретний тестовий приклад x та навчальну вибірку, і за характеристичну функцію безлічі ознак вважатимемо передбачення моделі, навченої лише на цих ознаках:

$$v(S) = f_S(x_S).$$

Отже, $\Delta(i, S)$ - зміна в передбаченні x між моделлю $f_{\{S \cup i\}}$, навченою на ознаках $S \cup \{i\}$ та моделлю f_S, навченою на ознаках S:

$$\Delta(i, S) = (f_{\{S \cup i\}}(x_{\{S \cup i\}} - f_S(x_S)).$$

Нехай ми маємо модель f, розподіл даних і тестовий приклад x і хочемо оцінити важливість поточних значень кожної ознаки порівняно з їх невизначеними значеннями. SHAP (SHapley Additive exPlanation) values для ознак на прикладі x - це Shapley values, що розраховуються для наступної кооперативної гри:

Ознаки у такому випадку виступають у ролі гравців (наявність i-го гравця відповідає поточному значенню i-ої ознаки з прикладу x, відсутність i-го гравця, відповідно, невизначеному значенню i-ої ознаки.

Характеристичною функцією $v(S)$ коаліції ознак S є умовна математичне очікування $E[f(x)|x_S]$ щодо розподілу даних.

Таким чином, алгоритм розрахунку SHAP values можна сформулювати так: для кожного можливого впорядкування ознак беруться всі ознаки, що стоять перед і-ю ознакою (позначимо їх за S) , а також

$$\Delta f(i, S) = E[f(x) \mid x\{S \cup i\}] - E[f(x) \mid x_S],$$

Наступний крок: усереднення отриманих значень за всіма упорядкуваннями. Іншими словами, SHAP values потрібні для описання очікуваного приросту вихідного значення моделі при додаванні і-ої ознаки в поточному прикладі.

Розділ 4. Порівняння результатів

4.1 Точність результатів

На наступних зображеннях можна побачити значення похибки для XGBoost (рис. 4.1а), LightGBM (рис. 4.1б), Catboost (рис.4.1в).

а)

```
Mean absolute error: 394.94338325825754
Mean squared error: 242254.01364245202
Mean absolute percentage error: 82.07441815015
```

б)

```
Mean absolute error: 192.15533269976376
Mean squared error: 77801.84733043819
Mean absolute percentage error: 28.127685180730
```

в)

```
Mean absolute error: 197.5592199080853
Mean squared error: 85374.9413442903
Mean absolute percentage error: 27.253702395887373
```

Рисунок 4.1. Значення похибки для XGBoost, LightGBM, Catboost.

Неважко помітити, що результати XGBoost сильно поступаються відповідним значенням для LightGBM та Catboost. Надати точну відповідь, що саме стало причиною такого результату для XGBoost, доволі важко, адже найбільш імовірне пояснення цього – сукупність факторів, таких як, оновлення самої бібліотеки XGBoost, недостатній рівень адаптації програми (частково виток з першого фактору) та, врешті, сам датасет – навіть після аналізу та підготовки даних, їх структура залишається доволі складною (чи, можна сказати, «незручною»), що, до речі, також несе відповідальність за погіршення відповідних результати решти моделей (LigthGBM та Catboost).

Зважаючи на отриману точність моделі XGBoost у 18% прийнято рішення не розглядати її у подальшому аналізі за допомогою механізму SHAP.

LigthGBM та Catboost: моделі показали приблизно однакову точність результату у прогнозуванні цін на автострахування, хоча точність Catboost, все ж таки, трохи перевищує значення відповідного показника у свого, так би

мовити, основного конкурента в цій роботі - LigthGBM (Mean absolute percentage error: 27.2537% у Catboost`а проти 28.12768% у LigthGBM). Така точність визнається прийнятною з урахуванням супутніх факторів, до того ж, порівняння результатів відбувається між моделями з практично однаковою точністю.

4.2 Аналіз SHAP

На зображенні можна побачити SHAP Summary Plot для випадку роботи:

1) Роботи з бібліотекою LightGBM (рис. 4.2а).
2) Роботи з бібліотекою CatBoost (рис. 4.2б).

а) б)

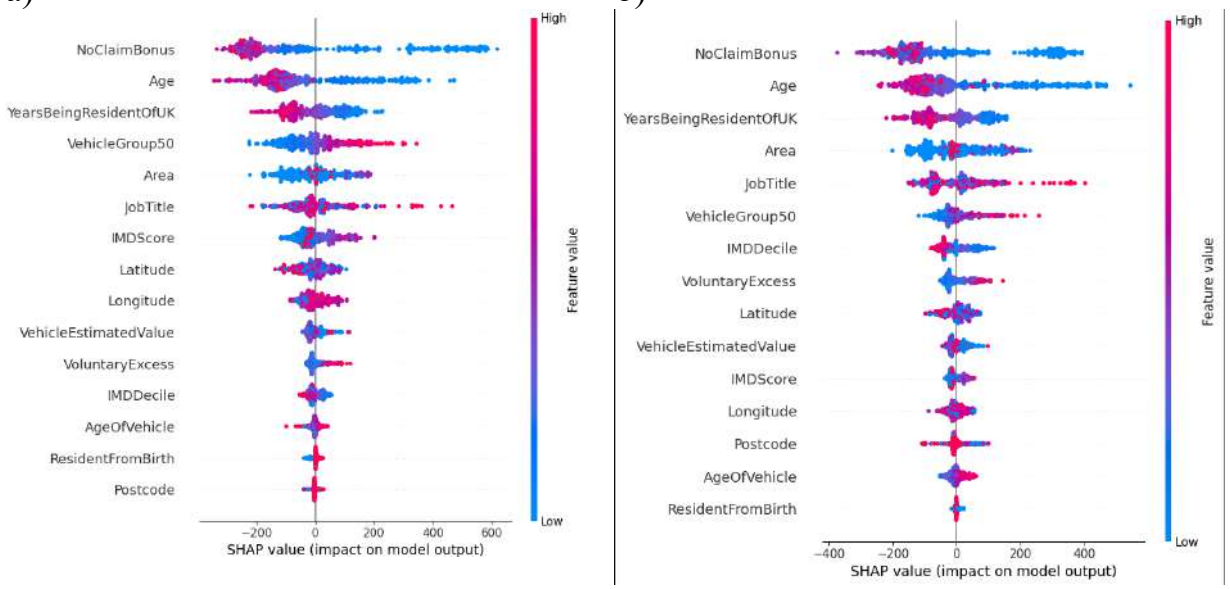


Рисунок 4.2. SHAP Summary Plot

Хоча на перший погляд структура двох моделей відрізняється, це не зовсім так.

Можна помітити, що порядок подання ознак для двох моделей відрізняється. На Summary plot`і ознаки розташовані у порядку «важливості», тобто за величиною абсолютного середнього значення SHAP. У якості прикладу можна привести 3 ознаки: Area, JobTitle та VehicleGroup50. Важливість ознаки VehicleGroup50 для моделі LightGBM є вищою, ніж важливості ознак Area та JobTitle (Area > JobTitle), у той час, коли для моделі Catboost важливість цієж ознаки оцінюється нижче, ніж як Area, так і JobTitle, при чому останні 2 ознаки розташовані ідентично першій моделі відносно один одного. Проте порядок розташування на Summary plot`і вказує на кількісну характеристику ознак.

На якісному рівні можна помітити практично ідентичну внутрішню структуру ознак. Проте і на цьому рівні є певні відмінності, навіть серед найважливіших ознак, таких як NoClaimBonus та Age .

а) LightGBM б) CatBoost

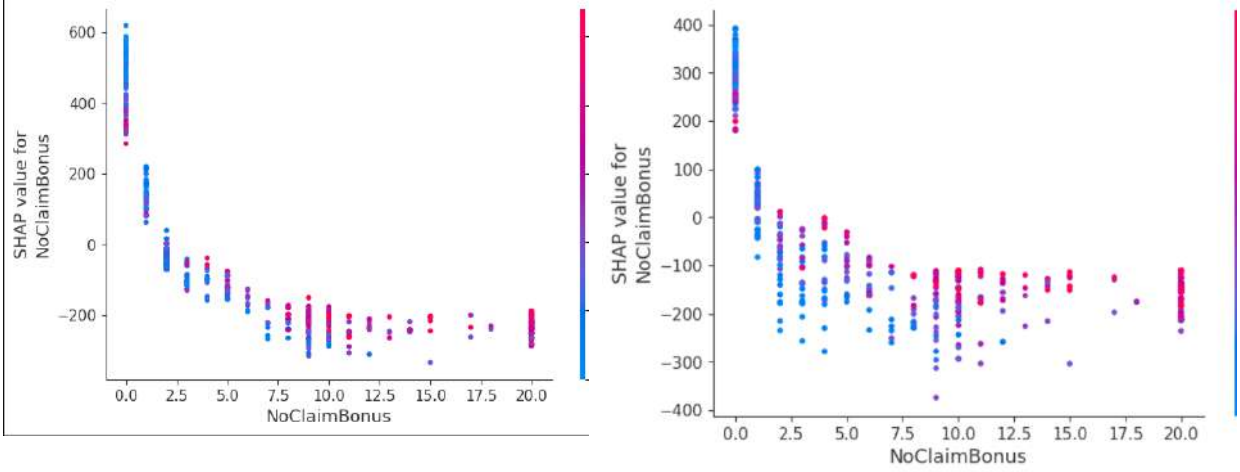


Рисунок 4.3. Dependence plot (NoClaimBonus)

На (рис. 4.3) зображені dependence plot'и з бібліотеки SHAP. У якості досліджуємої ознаки виступає NoClaimBonus, найвпливовіша ознака для обох моделей. У якості найбільш пов'язаної до неї ознаки - Age. Таким шляхом можна під іншим кутом побачити ту саму різницю між інтерпретаціями моделями ознаки,

Такі графіки можуть бути побудовані для будь якої ознаки (комбінації ознак), за замовчуванням обирається комбінація з найбільш тісно пов'язаною з решти ознак. На цьому зображенні розглядаються dependence plot`и для ознаки NoClaimBonus у парі з ознакою Age. Це дозволяє наочно побачити різницю між інтерпретацією моделями конкретних ознак та впливу цього на загальний результат моделі.

На рис. 4.4 також dependence plot'и для двох моделей, ознака представляється також у комбінації з найтісніше пов'язаною ознакою Age, але цього разу цей графік побудовано для ознаки VehicleEstimatedValue.

а) LightGBM б) CatBoost

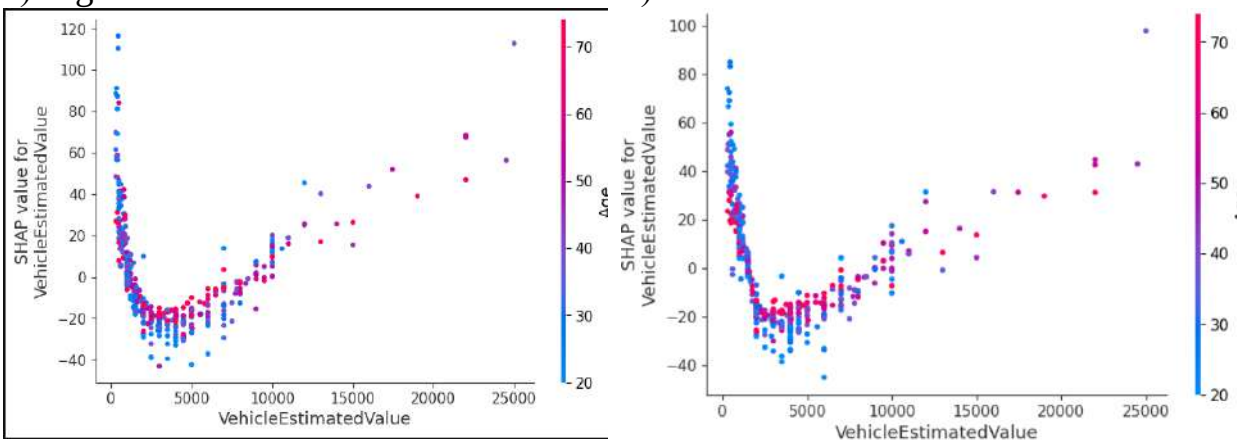


Рисунок 4.4. Dependence plot (VehicleEstimatedValue)

Хоч результати і дуже близькі один до одного, варто пам’ятати, що мова іде про масштабне використання, починаючи від об’єму вхідних даних і закінчуючи різноманітністю факторів, які впливають на результати моделі.

У цій роботі перевага надається моделі CatBoost у зв’язку з її більш універсальним набором інструментів та вищим, хоч і не значно, показником точності. До того ж, при збільшенні масштабів завдання саме CatBoost відрізняється точністю, у той час коли LightGBM все ж таки втрачає певну її частину.

Загалом, обидві моделі (LightGBM та CatBoost) продемонстрували очікувано гарний результат. Кожна з моделей має свої переваги та недоліки, які, в залежності від задачі та вхідних даних, можуть впливати на результати моделі та їх точність.

Розділ 5. Висновки

У роботі було проведено огляд різних механізмів як аналізу та підготовки даних, так і їх подальшого безпосереднього аналізу. Не дивлячись на усі можливі намагання підвищити точність результату моделей, вважається, що якогось значного результату у цьому аспекті досягнути не вдалось. Точність двох моделей приблизно однакова: похибка дорівнює 27% у моделі CatBoost та 28% у моделі LightGBM. Похибка результату третьої моделі (XGBoost) тримається на рівні 82%, що стало причиною не включення її у аналіз SHAP.

Моделі CatBoost та LightGBM мають однакові найважливіші ознаки, такі як NoClaimBonus, Age та YearsBeingResidentOfUK. Починаючи з четвертої ознаки по порядку важливості починаються деякі відмінності, але загальна структура залишається схожою. Були помічені як якісні, так і кількісні відмінності у оцінюванні моделями важливості ознак, що може дати поштовх до формування нових, або доповнення старих варіантів описання процесу механізмів машинного навчання для прогнозування цін на автострахування.

Список використаних джерел

[1] Li, Cheng. "A gentle introduction to gradient boosting." URL: http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf (2016): 30.

[2] Natekin, Alexey, and Alois Knoll. "Gradient boosting machines, a tutorial." Frontiers in neurorobotics 7 (2013): 21.

[3] Інтернет ресурс «Gradient Boosting Algorithm: A Complete Guide for Beginners». URL: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

[4] Natekin, Alexey, and Alois Knoll. "Gradient boosting machines, a tutorial." Frontiers in neurorobotics 7 (2013): 21.

[5] Su, Jiang, and Harry Zhang. "A fast decision tree learning algorithm." Aaai. Vol. 6. 2006.

[6] Priyam, Anuja, et al. "Comparative analysis of decision tree classification algorithms." International Journal of current engineering and technology 3.2 (2013): 334-337.

[7] Quinlan, J. Ross. "Learning decision tree classifiers." ACM Computing Surveys (CSUR) 28.1 (1996): 71-72.

[8] Brownlee, Jason. Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery, 2020.

[9] Barapatre, Darshan, and A. Vijayalakshmi. "Data preparation on large datasets for data science." Asian Journal of Pharmaceutical and Clinical Research 10.13 (2017): 485-488.

[10] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016. p. 785-794.

[11] Інтернет-ресурс – офіційна документація xgboost

URL: https://xgboost.readthedocs.io/en/stable/

[12] Friedman, Jerome H. "Stochastic gradient boosting." Computational statistics & data analysis 38.4 (2002): 367-378.

[13] Wang, Chen, Chengyuan Deng, and Suzhen Wang. "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost." Pattern Recognition Letters 136 (2020): 190-197.

[14] Torlay, L., et al. "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy." Brain informatics 4.3 (2017): 159-169

[15] Wang, Yan, and Xuelei Sherry Ni. "A XGBoost risk model via feature selection and Bayesian hyper-parameter optimization." arXiv preprint arXiv:1901.08433 (2019).

[16] Інтернет-ресурс "XGBoost Mathematics Explained" URL: https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a

[17] Інтернет-ресурс "Introduction to Taylor's theorem for multivariable functions"

URL: https://mathinsight.org/taylors_theorem_multivariable_introduction

[18] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in neural information processing systems 30 (2017).

[19] Machado, Marcos Roberto, Salma Karray, and Ivaldo Tributino de Sousa. "LightGBM: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry." 2019 14th International Conference on Computer Science & Education (ICCSE). IEEE, 2019.

[20] Hancock, John, and Taghi M. Khoshgoftaar. "Leveraging lightgbm for categorical big data." 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, 2021.

[21] Hancock, John T., and Taghi M. Khoshgoftaar. "CatBoost for big data: an interdisciplinary review." Journal of big data 7.1 (2020): 1-45.

[22] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems, 31.

[23] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363.

[24] Інтернет-ресурс “CatBoost regression in 6 minutes” URL: https://towardsdatascience.com/catboost-regression-in-6-minutes-3487f3e5b329

[25] Інтернет-ресурс “Интерпретация моделей и диагностика сдвига данных: LIME, SHAP и Shapley Flow” URL: https://habr.com/ru/companies/ods/articles/599573/

[26] Merrick, L., & Taly, A. (2020). The explanation game: Explaining machine learning models using shapley values. In Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4 (pp. 17-38). Springer International Publishing.

[27]Wang, J., Wiens, J., & Lundberg, S. (2021, March). Shapley flow: A graph-based approach to interpreting model predictions. In International Conference on Artificial Intelligence and Statistics (pp. 721-729). PMLR.

[28] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." Advances in neural information processing systems 30 (2017).

[29] Molnar, Christoph. Interpretable machine learning. Lulu. com, 2020.

[30] Parsa, Amir Bahador, et al. "Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis." Accident Analysis & Prevention 136 (2020): 105405.