Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Предсказание суммы трат пользователя в зависимости от его интересов

Студент **ИУ5-31М** _____ **Курганова А.Г.**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **Гапанюк Ю.Е.**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2021 г.

Цель работы

Целью этой научно-исследовательской работы является предсказание суммы трат пользователя, в зависимости от его интересов на основе данных от телекомпаний.

Введение

В 21 веке люди смотрят телевизор у себя дома каждый день. В нынешние время из-за пандемии очень много людей вынуждены оставаться дома ради своего здоровья. Находясь дома неделями, человек начинает искать способ себя развлечь и как правило его выбор падает на просмотр телевизора и разных программ по нему.

Из-за продолжительного пребывания дома телевизор иногда перестает быть таким интересным развлечением. Тогда многие люди начинают расширять программы, которые можно посмотреть путем множества подписок на каналы или же начинают платить телеоператору за другие дополнительные услуги, которые предоставляют больше возможностей для использования телевизора.

Решение задачи предсказания суммы трат пользователя, в зависимости от его интересов

Для начала датасет был исследован на наличие пропусков, затем было выполнено кодирование категориальных признаков, таких как 'gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn'.

После кодирования категориальных признаков. Было проведено обучение модели с различными вариантами, были построены графики метрик качества модели и модель с использованием AutoML.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from IPython.display import Image
import scipy.stats as stats
%matplotlib inline
sns.set(style="ticks")
```

```
data = pd.read_csv(path+'/telecom_users.csv')
```

```
data.head()
```

	Unnamed: 0	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
0	1869	7010-BRBUU	Male	0	Yes	Yes	72
1	4528	9688-YGXVR	Female	0	No	No	44
2	6344	9286-DOJGF	Female	1	Yes	No	38
3	6739	6994-KERXL	Male	0	No	No	4
4	432	2181-UAESM	Male	0	No	No	2

```
data.shape
```

```
(5986, 22)
```

Пропуски

```
data_features = list(zip(
# признаки
[i for i in data.columns],
zip(
    # типы колонок
    [str(i) for i in data.dtypes],
    # проверим есть ли пропущенные значения
    [i for i in data.isnull().sum()]
)))
# Признаки с типом данных и количеством пропусков
data_features
```

```
[('Unnamed: 0', ('int64', 0)),
 ('customerID', ('object', 0)),
 ('gender', ('object', 0)),
 ('SeniorCitizen', ('int64', 0)),
 ('Partner', ('object', 0)),
 ('Dependents', ('object', 0)),
 ('tenure', ('int64', 0)),
 ('PhoneService', ('object', 0)),
 ('MultipleLines', ('object', 0)),
 ('InternetService', ('object', 0)),
 ('OnlineSecurity', ('object', 0)),
 ('OnlineBackup', ('object', 0)),
 ('DeviceProtection', ('object', 0)),
 ('TechSupport', ('object', 0)),
 ('StreamingTV', ('object', 0)),
 ('StreamingMovies', ('object', 0)),
 ('Contract', ('object', 0)),
 ('PaperlessBilling', ('object', 0)),
 ('PaymentMethod', ('object', 0)),
 ('MonthlyCharges', ('float64', 0)),
 ('TotalCharges', ('object', 0)),
 ('Churn', ('object', 0))]
```

Пропусков в датасете не обнаружено.

Кодирование категориальных признаков

```
from sklearn.preprocessing import LabelEncoder
```

```
data_encoded = data.copy()


columns = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', '
for column in columns:
    label_encoder = LabelEncoder()
    data_encoded[column] = label_encoder.fit_transform(data_encoded[column])


data_encoded = data_encoded.drop([data_encoded.columns[0], data_encoded.columns[
data_encoded.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multiple
0	1	0	1	1	72	1	
1	0	0	0	0	44	1	
2	0	1	1	0	38	1	
3	1	0	0	0	4	1	
4	1	0	0	0	2	1	

```
from sklearn.model_selection import train_test_split
```

```
x = data_encoded.drop([data_encoded.columns[-1]], axis=1)
x
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multi
0	1	0	1	1	72	1	
1	0	0	0	0	44	1	
2	0	1	1	0	38	1	
3	1	0	0	0	4	1	
4	1	0	0	0	2	1	
...	...	...	...	...	...	...	...
5981	1	0	1	0	1	1	
5982	0	0	1	1	23	1	
5983	1	0	1	1	12	1	
5984	1	1	0	0	12	1	
5985	1	0	0	0	26	1	

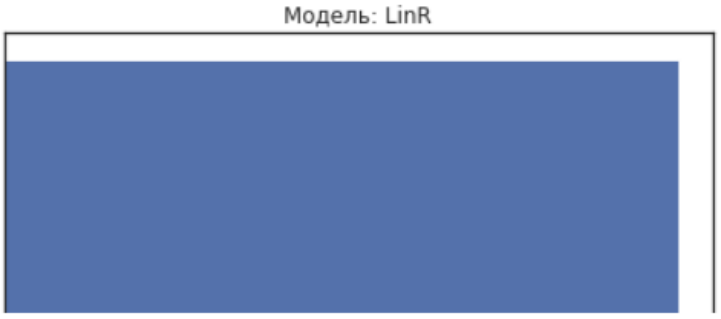
5986 rows × 17 columns

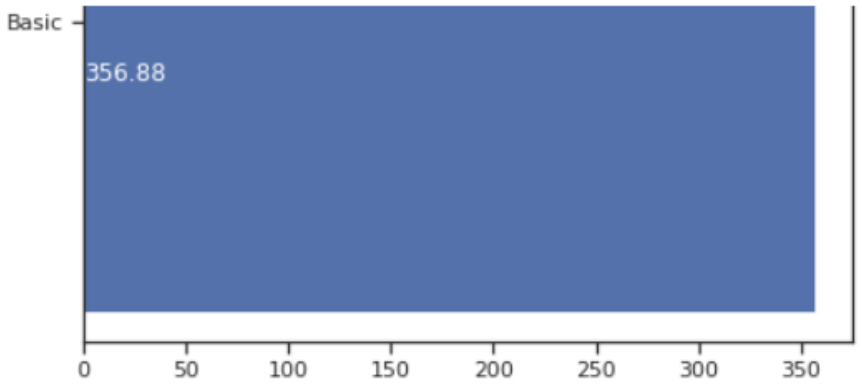
```
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=x.columns)
    return res


# Разделим выборку на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(x, data_encoded['MonthlyChar
                                                    test_size=0.2,
                                                    random_state=1)
# Преобразуем массивы в DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape, y_train.shape, y_test.shape
```

```
((4788, 17), (1198, 17), (4788,), (1198,))
```

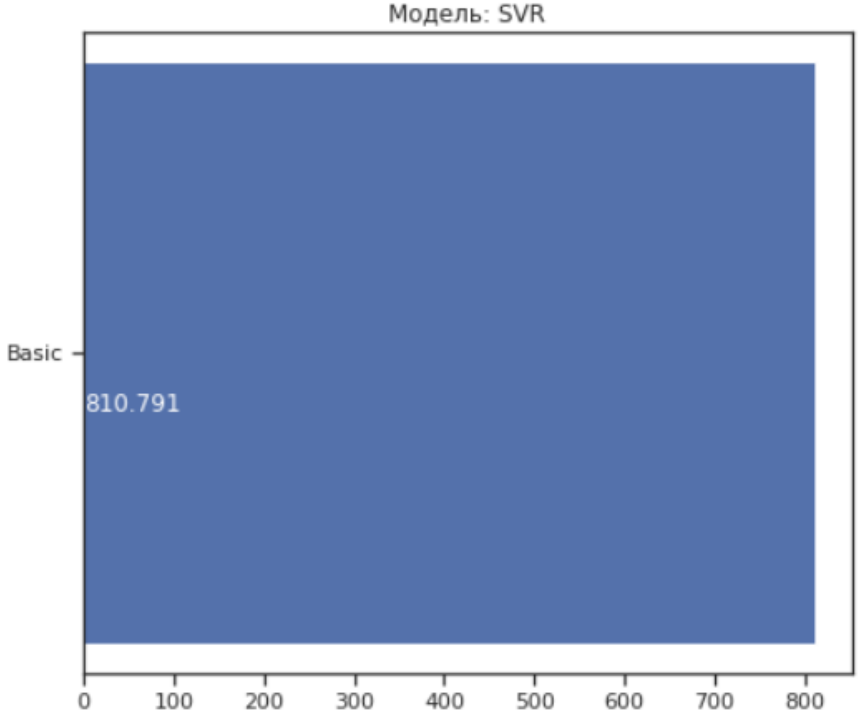
```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascendin
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()
```

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error


clas_models_dict = {'LinR': LinearRegression(),
                    'SVR': SVR(),
                    'KNN_5':KNeighborsRegressor(n_neighbors=5),
                    'Tree':DecisionTreeRegressor(random_state=1),
                    'GB': GradientBoostingRegressor(random_state=1),
                    'RF':RandomForestRegressor(n_estimators=50, random_state=1


X_data_dict = {'Basic': (X_train_df, X_test_df)}


def test_models(clas_models_dict, X_train, X_test, y_train, y_test):

    logger = MetricLogger()

    for model_name, model in clas_models_dict.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        logger.add(model_name, 'Basic', mse)

    return logger


logger = test_models(clas_models_dict, X_train_df, X_test_df, y_train, y_test)


# Построим графики метрик качества модели
for model in clas_models_dict:
    logger.plot('Модель: ' + model, model, figsize=(7, 6))
```

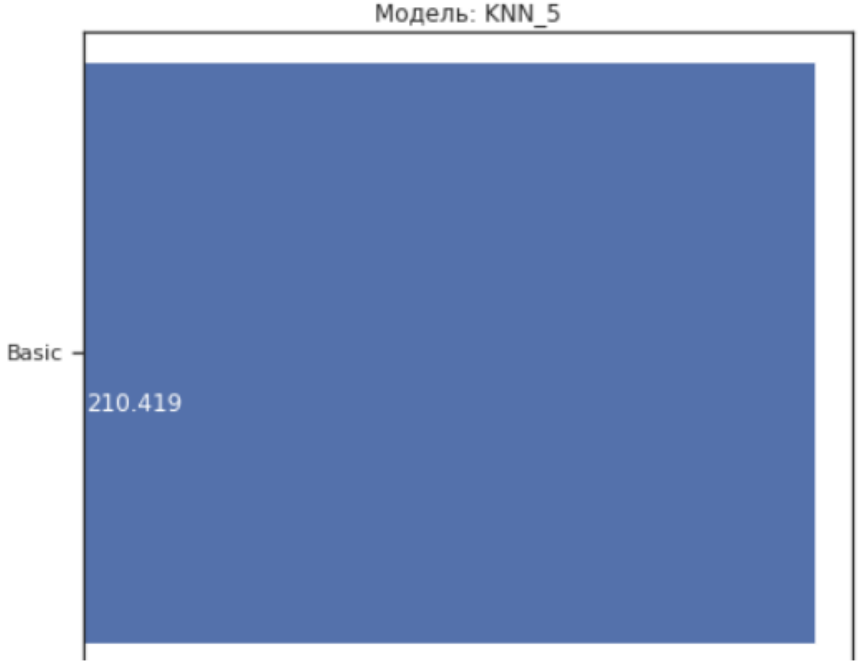


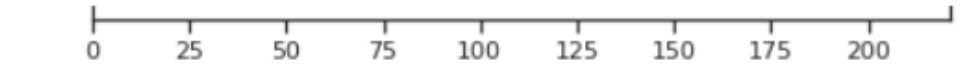


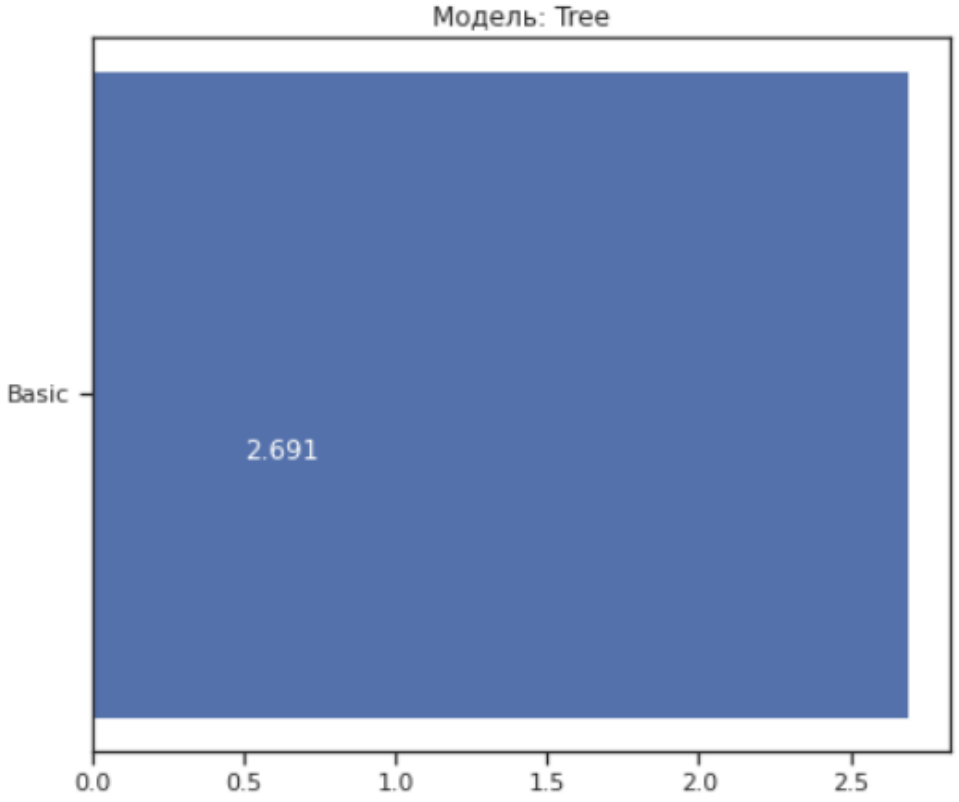
Модель: SVR

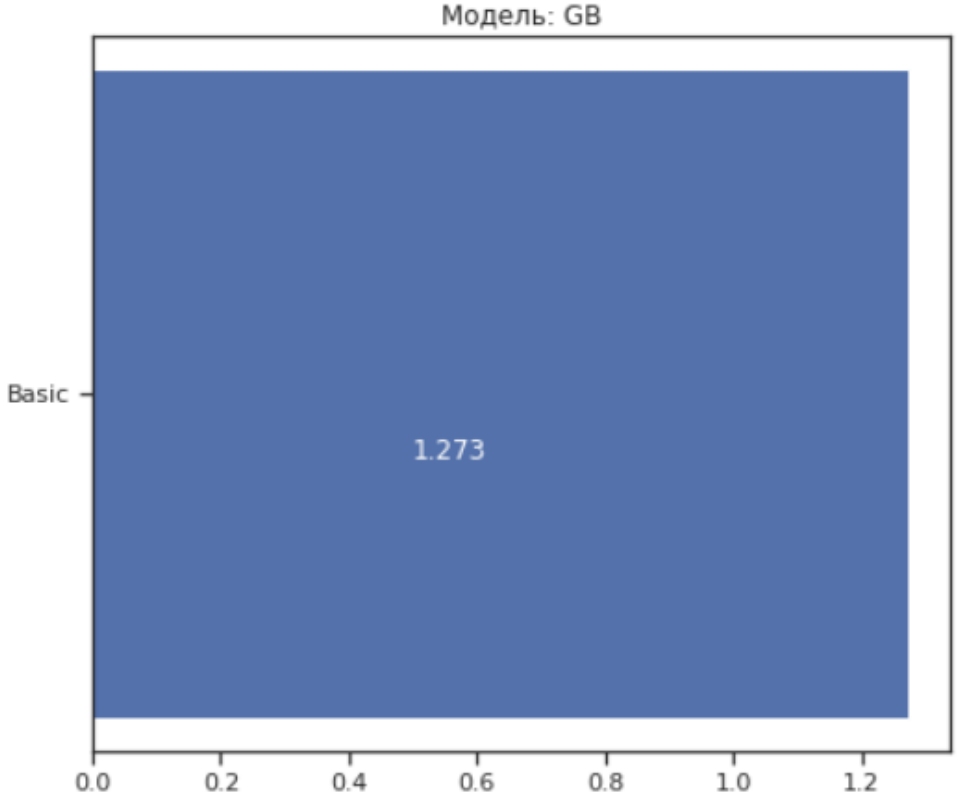


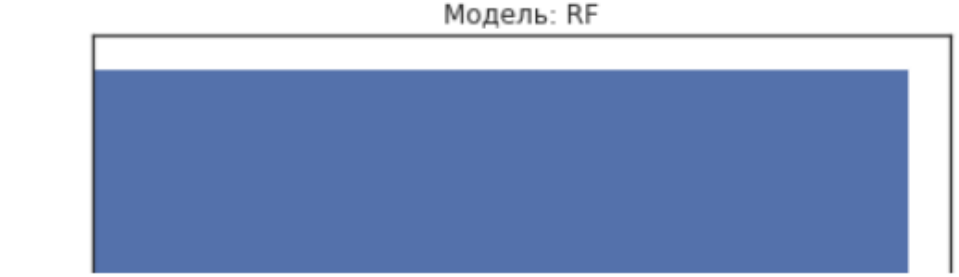
Модель: KNN_5

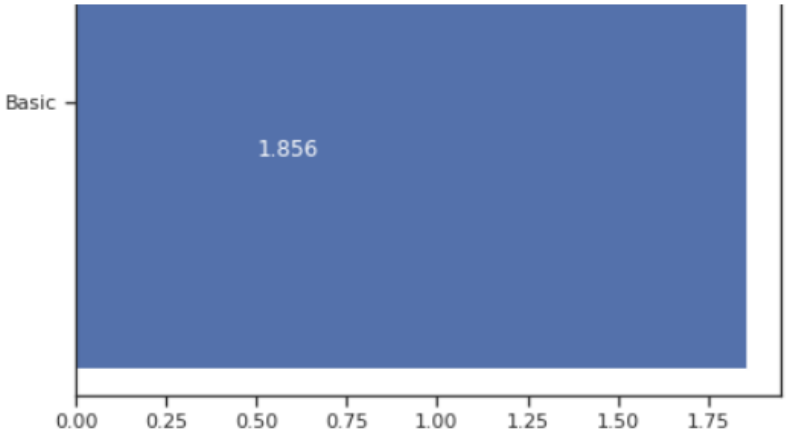



0
25
50
75
100
125
150
175
200


Модель: Tree
Basic
2.691
0.0
0.5
1.0
1.5
2.0
2.5


Модель: GB
Basic
1.273
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Модель: RF



AutoML

```
!pip3 install --user mljar-supervised
!pip3 install delayed
```

```
Requirement already satisfied: mljar-supervised in /root/.local/lib/python3
Requirement already satisfied: category-encoders==2.2.2 in /usr/local/lib/p
Requirement already satisfied: wordcloud==1.8.1 in /root/.local/lib/python3
Requirement already satisfied: tabulate==0.8.7 in /root/.local/lib/python3.
Requirement already satisfied: cloudpickle==1.3.0 in /usr/local/lib/python3
Requirement already satisfied: scikit-plot==0.3.7 in /root/.local/lib/pythc
Requirement already satisfied: dtreeviz==1.3 in /root/.local/lib/python3.7/
Requirement already satisfied: scikit-learn==0.24.2 in /root/.local/lib/pyt
Requirement already satisfied: catboost==0.24.4 in /root/.local/lib/python3
Requirement already satisfied: pandas==1.2.0 in /root/.local/lib/python3.7/
Requirement already satisfied: markdown in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: xgboost==1.3.3 in /root/.local/lib/python3.7
Requirement already satisfied: lightgbm==3.0.0 in /root/.local/lib/python3.
Requirement already satisfied: optuna==2.7.0 in /root/.local/lib/python3.7/
Requirement already satisfied: seaborn==0.11.1 in /usr/local/lib/python3.7/
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.
Requirement already satisfied: joblib==1.0.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyarrow>=2.0.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: scipy==1.6.1 in /root/.local/lib/python3.7/s
Requirement already satisfied: numpy>=1.20.0 in /root/.local/lib/python3.7/
Requirement already satisfied: shap==0.36.0 in /root/.local/lib/python3.7/s
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pytest in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: colour in /root/.local/lib/python3.7/site-pa
Requirement already satisfied: graphviz>=0.9 in /usr/local/lib/python3.7/di
Requirement already satisfied: threadpoolctl>=2.0.0 in /root/.local/lib/pyt
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-package
```

```
train = data
train.head()
```

	Unnamed: 0	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
0	1869	7010-BRBUU	Male	0	Yes	Yes	72
1	4528	9688-YGXVR	Female	0	No	No	44
2	6344	9286-DOJGF	Female	1	Yes	No	38
3	6739	6994-KERXL	Male	0	No	No	4
4	432	2181-UAESM	Male	0	No	No	2

```
automl = AutoML()
```

```
automl.fit(train[train.columns[2:-3]], train['MonthlyCharges'])
```

```
AutoML directory: AutoML_1
The task is regression with evaluation metric rmse
AutoML will use algorithms: ['Baseline', 'Linear', 'Decision Tree', 'Random
AutoML will ensemble availabe models
AutoML steps: ['simple_algorithms', 'default_algorithms', 'ensemble']
* Step simple_algorithms will try to check up to 3 models
1_Baseline rmse 30.041484 trained in 0.42 seconds
2_DecisionTree rmse 9.566765 trained in 13.15 seconds
3_Linear rmse 36.208657 trained in 3.66 seconds
* Step default_algorithms will try to check up to 3 models
4_Default_Xgboost rmse 1.103866 trained in 7.6 seconds
5_Default_NeuralNetwork rmse 2.882078 trained in 1.61 seconds
6_Default_RandomForest rmse 6.037609 trained in 9.99 seconds
* Step ensemble will try to check up to 1 model
Ensemble rmse 1.103866 trained in 0.37 seconds
/root/.local/lib/python3.7/site-packages/numpy/lib/function_base.py:2642: R

invalid value encountered in true_divide

/root/.local/lib/python3.7/site-packages/numpy/lib/function_base.py:2643: R

invalid value encountered in true_divide

AutoML fit time: 49.05 seconds
AutoML best model: 4_Default_Xgboost
AutoML()
```