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_____

КАФЕДРА _____Системы обработки информации и управления_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

ПО КУРСУ:

Технологии машинного обучения

Студент _____ИУ5-62_____ _____ _____**Муравьев О.М.**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ ____**Гапанюк Ю.Е.**___
(Подпись, дата) (И.О.Фамилия)

2019 г.

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)

З А Д А Н И Е
на выполнение курсовой работы

1. Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
8. Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
9. Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
11. Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется постройение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Введение

Курсовая работа – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсовой работы является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовая работа опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.

Содержание

Основная часть

Описание набора данных

Набор данных представляет собой множество признаков для описания домов. Цель: на основе признаков предсказать стоимость дома.

Подключение библиотек

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error as MAE, median_absolute_error as MedAE, r2_score as R2
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
```

Подготовка набора данных к работе

```
print(data.shape)
```

```
(1460, 81)
```

Типы данных

```
data_types={'float64':0, 'object':0, 'int64':0}
for i in data.dtypes:
    data_types[str(i)]+=1
```

```
float64 3
object 43
int64 35
```

Заполним пропуски, но удалим столбцы, где пропусков больше половины:

```
nd = data.columns[data.isnull().any()]
catsnnd = [col for col in nd if data[col].dtype=="object"]
print(catsnnd)
```

```
['Alley', 'MasVnrType', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Electrical', 'Firepl
aceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature']
```

```
catsnnd.remove('MiscFeature')
catsnnd.remove('Fence')
catsnnd.remove('PoolQC')
```

```
for column in catsnnd:
    temp_data = data[[column]]
    imp = SimpleImputer(strategy='most_frequent')
    data_num_imp = imp.fit_transform(temp_data)
    data[[column]] = data_num_imp
```

```
for column in intnd:
    temp_data = data[[column]]
    imp = SimpleImputer(strategy='median')
    data_num_imp = imp.fit_transform(temp_data)
    data[[column]] = data_num_imp
```

```
data=data.dropna(axis=1, how='any')
```

```
nd = data.columns[data.isnull().any()]
print(len(nd))
```

```
0
```

Закодируем категориальные признаки

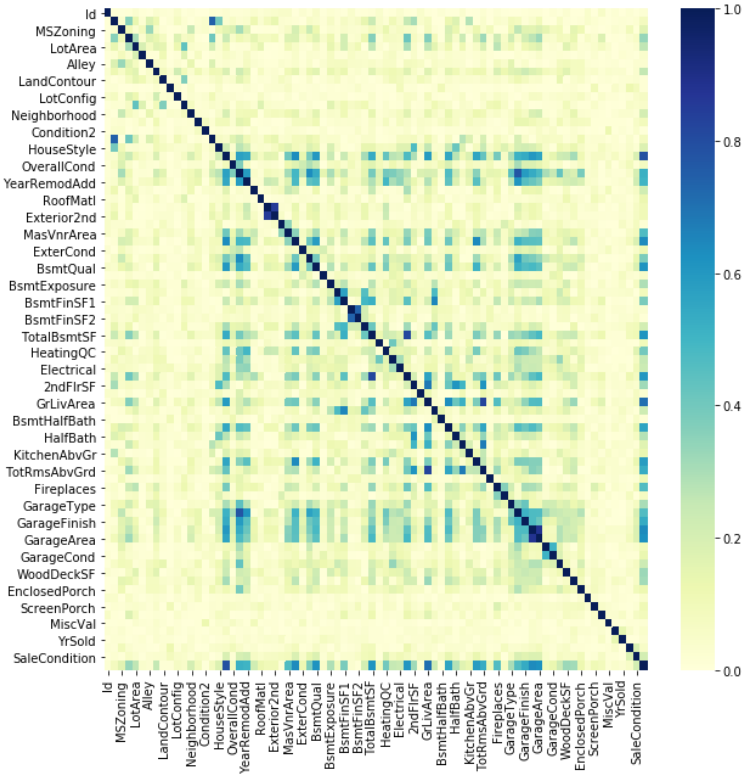
```
cats = [col for col in data.columns if data[col].dtype=="object"]
le = LabelEncoder()
for col in cats:
    data[col]=le.fit_transform(data[col])
cats = [col for col in data.columns if data[col].dtype=="object"]
print(len(cats))
```

```
0
```

Матрица корреляции

```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(data.corr().abs(),cmap='YlGnBu', annot=False, fmt='.3f', ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1331080>
```



Посмотрим значения корреляции признаков по отношению к целевому

```
data.corr()['SalePrice'].abs().sort_values(ascending=False)
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
ExterQual       0.636884
GarageArea      0.623431
BsmtQual        0.618025
TotalBsmtSF     0.613581
1stFlrSF        0.605852
KitchenQual     0.589189
FullBath        0.560664
GarageFinish    0.537242
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
MasVnrArea      0.472614
Fireplaces      0.466929
GarageYrBlt     0.466754
HeatingQC       0.400178
BsmtFinSF1      0.386420
Foundation      0.382479
```

Оставим только те признаки, у которых коэффициент корреляции больше 0.5

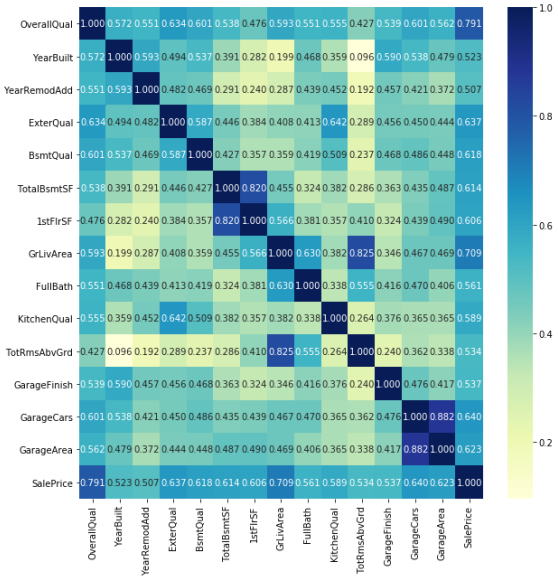
```
good_corr = []
for k, v in (dict(data.corr()['SalePrice'].abs())).items():
    if v >=0.5:
        good_corr.append(k)

print(good_corr)
```

```
['OverallQual', 'YearBuilt', 'YearRemodAdd', 'ExterQual', 'BsmtQual', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'FullBa
th', 'KitchenQual', 'TotRmsAbvGrd', 'GarageFinish', 'GarageCars', 'GarageArea', 'SalePrice']
```

```
good_corr_data = data[good_corr]
print(good_corr_data.columns)
```

```
Index(['OverallQual', 'YearBuilt', 'YearRemodAdd', 'ExterQual', 'BsmtQual',
       'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'FullBath', 'KitchenQual',
       'TotRmsAbvGrd', 'GarageFinish', 'GarageCars', 'GarageArea',
       'SalePrice'],
      dtype='object')
```

Посмотрим на получившуюся карту корреляции и удалим признаки, которые сильно коррелируют друг с другом, но слабо с целевым признаком.

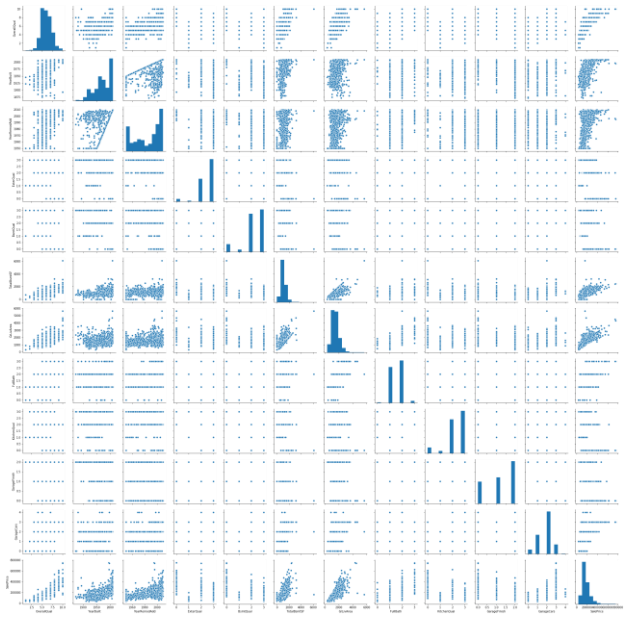



```
good_corr.remove('TotRmsAbvGrd')
good_corr.remove('GarageArea')
good_corr.remove('1stFlrSF')
```

```
good_corr_data = data[good_corr]
print(good_corr_data.columns)
```

```
Index(['OverallQual', 'YearBuilt', 'YearRemodAdd', 'ExterQual', 'BsmtQual',
       'TotalBsmtSF', 'GrLivArea', 'FullBath', 'KitchenQual', 'GarageFinish',
       'GarageCars', 'SalePrice'],
      dtype='object')
```

Парные диаграммы для всего получившегося набора данных



Выбраны метрики:

- 1 - R2 метрика - самый лучший выбор в задачи регрессии
- 2 - MAE - средняя абсолютная ошибка, преимущество его в том оно не сильно штрафует за выбросы в данных
- 3 - MedAE - среднее медианное отклонение, показывает отклонение от средней величины

Выбраны модели:

- 1 - RandomForest - ансамблевая модель Случайный лес - один из немногих универсальных алгоритмов
- 2 - Desicion Tree - дерево решений, простая и наглядная модель.
- 3 - SVM - наиболее быстрый метод нахождения решающих функций
- 4 - GradientBoosting - градиентный бустинг, на сегодняшний день является одним из самых мощных алгоритмов распознавания.
- 5 - ElasticNet - простая линейная модель, с регуляризацией

Разделим выборку на тренировочную и тестовую

```
y=data["SalePrice"]
```

```
x= x=data.drop("SalePrice",axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Обучение моделей и их проверка

RandomForest

```
RF=RandomForestRegressor().fit(X_train, y_train)
```

```
/Users/lina/Documents/ML-Technology/.venv/lib/python3.7
The default value of n_estimators will change from 10 i
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
res_RF = RF.predict(X_test)
print(MAE(y_test,res_RF))
print(MedAE(y_test,res_RF))
print(R2(y_test,res_RF))
```

```
18587.838356164386
12286.299999999988
0.8934839971343332
```

Desicion Tree

```
DT=DecisionTreeRegressor().fit(X_train, y_train)
```

```
res_DT = DT.predict(X_test)
print(MAE(y_test,res_DT))
print(MedAE(y_test,res_DT))
print(R2(y_test,res_DT))
```

```
27939.921232876713
17068.0
0.7659685028584341
```

Desicion Tree

```
DT=DecisionTreeRegressor().fit(X_train, y_train)
```

```
res_DT = DT.predict(X_test)
print(MAE(y_test,res_DT))
print(MedAE(y_test,res_DT))
print(R2(y_test,res_DT))
```

```
27939.921232876713
17068.0
0.7659685028584341
```

GradientBoosting

```
GB=GradientBoostingRegressor(
    n_estimators=10,
    max_depth=10,
    learning_rate=0.01).fit(X_train, y_train)
```

```
res_GB = GB.predict(X_test)
print(MAE(y_test,res_GB))
print(MedAE(y_test,res_GB))
print(R2(y_test,res_GB))
```

```
57224.84342146683
44483.88575610731
0.1529978466864158
```

ElasticNet

```
EN=ElasticNet().fit(X_train, y_train)
```

```
/Users/lina/Documents/ML-Technology/.venv/lib
ConvergenceWarning: Objective did not converg
h very small alpha may cause precision proble
  ConvergenceWarning)
```

```
res_EN = EN.predict(X_test)
print(MAE(y_test,res_EN))
print(MedAE(y_test,res_EN))
print(R2(y_test,res_EN))
```

```
20194.390884847497
12222.416664172197
0.8467463664681263
```

Подбор гиперпараметров и обучение моделей с их использованием

RandomForest

```
param = {'n_estimators':range(50,200,10)}
new_RF = GridSearchCV(RandomForestRegressor(), param, cv=5,scoring='r2')
new_RF.fit(X_train, y_train)
```

```
new_RF.best_params_
```

```
{'n_estimators': 80}
```

```
new_RF_reg=RandomForestRegressor(n_estimators=80).fit(X_train,y_train)
```

```
new_res_RF = new_RF_reg.predict(X_test)
print(f'По умолчанию: {MAE(y_test,res_RF)}, с гиперпараметром: {MAE(y_test,new_res_RF)}')
print(f'По умолчанию: {MedAE(y_test,res_RF)}, с гиперпараметром: {MedAE(y_test,new_res_RF)}')
print(f'По умолчанию: {R2(y_test,res_RF)}, с гиперпараметром: {R2(y_test,new_res_RF)}')
```

```
По умолчанию: 18587.838356164386, с гиперпараметром: 17787.38056506849
По умолчанию: 12286.299999999988, с гиперпараметром: 10616.506250000006
По умолчанию: 0.8934839971343332, с гиперпараметром: 0.8961845839878957
```

DecisionTree

```
param = {'min_impurity_decrease':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}
new_DT = GridSearchCV(DecisionTreeRegressor(), param, cv=5,scoring='r2')
new_DT.fit(X_train, y_train)
```

```
new_DT.best_params_
```

```
{'min_impurity_decrease': 0.1}
```

```
new_DT_reg = DecisionTreeRegressor(min_impurity_decrease= 0.1).fit(X_train, y_train)
```

```
new_res_DT = new_DT_reg.predict(X_test)
print(f'По умолчанию: {MAE(y_test,res_DT)}, с гиперпараметром: {MAE(y_test,new_res_DT)}')
print(f'По умолчанию: {MedAE(y_test,res_DT)}, с гиперпараметром: {MedAE(y_test,new_res_DT)}')
print(f'По умолчанию: {R2(y_test,res_DT)}, с гиперпараметром: {R2(y_test,new_res_DT)}')
```

```
По умолчанию: 27939.921232876713, с гиперпараметром: 26384.325342465752
По умолчанию: 17068.0, с гиперпараметром: 15500.0
По умолчанию: 0.7659685028584341, с гиперпараметром: 0.7770317689158205
```

SVR

```
param = {'gamma':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1],'C':[1,10,100,1000]}
new_SVR = GridSearchCV(SVR(), param, cv=5,scoring='r2')
new_SVR.fit(X_train, y_train)
```

```
new_SVR.best_params_
```

```
{'C': 1000, 'gamma': 0.9}
```

```
new_SVR_reg = SVR(gamma = 0.9, C= 1000).fit(X_train, y_train)
```

```
new_res_SVR = new_SVR_reg.predict(X_test)
print(f'По умолчанию: {MAE(y_test,res_svm)}, с гиперпараметром: {MAE(y_test,new_res_SVR)}')
print(f'По умолчанию: {MedAE(y_test,res_svm)}, с гиперпараметром: {MedAE(y_test,new_res_SVR)}')
print(f'По умолчанию: {R2(y_test,res_svm)}, с гиперпараметром: {R2(y_test,new_res_SVR)}')
```

```
По умолчанию: 59568.202511415526, с гиперпараметром: 59560.74095890412
По умолчанию: 40000.43333333332, с гиперпараметром: 40050.0
По умолчанию: -0.0249731726987783, с гиперпараметром: -0.025219486615010167
```

GradientBoosting

```
param = {'n_estimators':[1, 3, 5, 7, 10], 'max_depth':[1, 3, 5, 7, 10],
         'learning_rate':[0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]}
new_GB = GridSearchCV(GradientBoostingRegressor(), param, cv=5,scoring='r2')
new_GB.fit(X_train, y_train)
```

```
new_GB.best_params_
```

```
{'learning_rate': 0.025, 'max_depth': 10, 'n_estimators': 10}
```

```
new_GB_reg=GradientBoostingRegressor(
    learning_rate = 0.025,
    max_depth = 10,
    n_estimators = 10).fit(X_train, y_train)
```

```
new_res_GB = new_GB_reg.predict(X_test)
print(f'По умолчанию: {MAE(y_test,res_GB)}, с гиперпараметром: {MAE(y_test,new_res_GB)}')
print(f'По умолчанию: {MedAE(y_test,res_GB)}, с гиперпараметром: {MedAE(y_test,new_res_GB)}')
print(f'По умолчанию: {R2(y_test,res_GB)}, с гиперпараметром: {R2(y_test,new_res_GB)}')
```

```
По умолчанию: 57224.84342146683, с гиперпараметром: 50099.35506393144
По умолчанию: 44483.88575610731, с гиперпараметром: 39600.43904238139
По умолчанию: 0.1529978466864158, с гиперпараметром: 0.3358876358716616
```

ElasticNet

```
param = {"max_iter": [1, 5, 10],
         "alpha": [0.0001, 0.001, 0.01, 0.1, 1, 10, 100],
         "l1_ratio": np.arange(0.0, 1.0, 0.1)}
new_EN = GridSearchCV(ElasticNet(), param, cv=10,scoring='r2')
new_EN.fit(X_train, y_train)
new_EN.best_params_
```

```
{'alpha': 1, 'l1_ratio': 0.8, 'max_iter': 10}
```

```
new_EN_reg=ElasticNet(alpha = 1,
                      l1_ratio = 0.8,
                      max_iter = 10).fit(X_train, y_train)
```

```
new_res_EN = new_EN_reg.predict(X_test)
print(f'По умолчанию: {MAE(y_test,res_EN)}, с гиперпараметром: {MAE(y_test,new_res_EN)}')
print(f'По умолчанию: {MedAE(y_test,res_EN)}, с гиперпараметром: {MedAE(y_test,new_res_EN)}')
print(f'По умолчанию: {R2(y_test,res_EN)}, с гиперпараметром: {R2(y_test,new_res_EN)}')
```

```
По умолчанию: 20194.390884847497, с гиперпараметром: 20576.613824233984
По умолчанию: 12222.416664172197, с гиперпараметром: 13448.601327728713
По умолчанию: 0.8467463664681263, с гиперпараметром: 0.8535237360191417
```

Вывод

В результате выполнения курсовой работы я обобщил знания, полученные на лекциях и при выполнении лабораторных работ. Я научился работать с ансамблевыми моделями и подбирать для них гиперпараметры. Личный вывод: ансамблевые модели во многом превосходят обычные. При этом для каждой задачи лучше подойдет определенный метод и определенная метрика.

На этом наборе данных хорошие результаты показали модели «Случайный лес» и «Градиентный бустинг».

Список литературы

1. Лекции по курсу. [Электронный ресурс]. Гапанюк Ю.Е. Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/COURSE_TMO
2. Специализация Машинное обучение и анализ данных. Coursera.org. [Электронный ресурс]. Режим доступа: https://www.coursera.org/specializations/machine-learning-data-analysis
3. Intermediate Machine Learning. Kaggle.com. [Электронный ресурс]. Режим доступа: https://www.kaggle.com/learn/intermediate-machine-learning