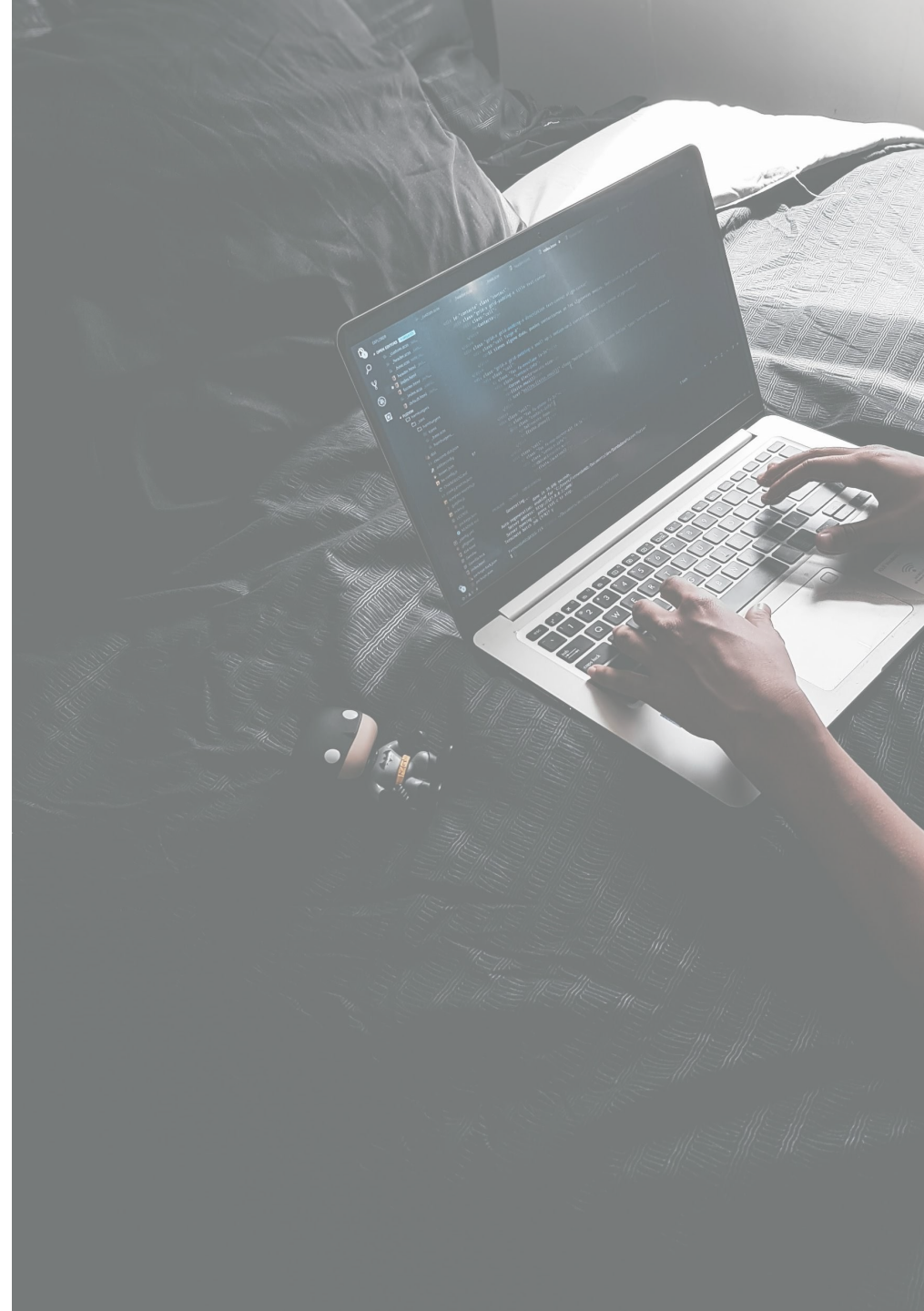
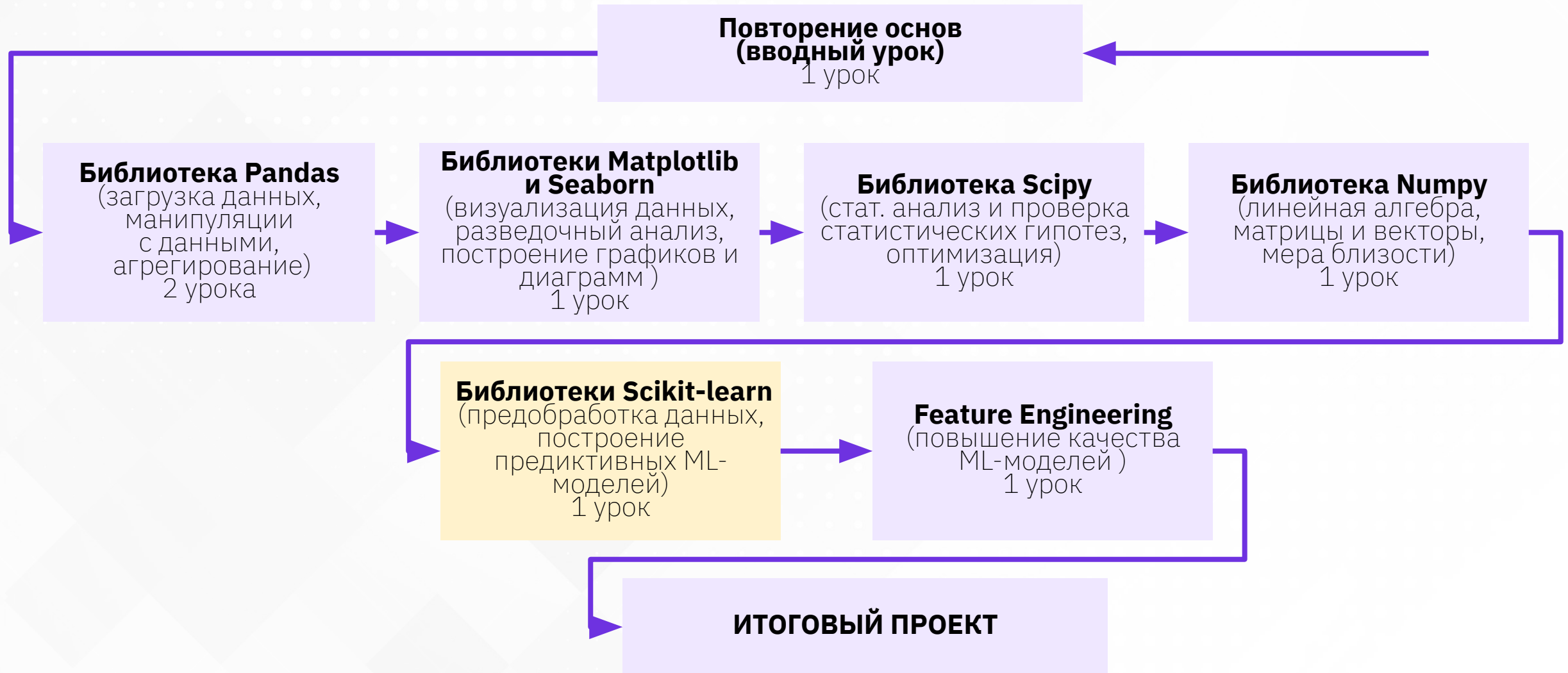


# Предобработка данных и построение предиктивных моделей в Scikit-learn

Python для аналитиков / урок 7



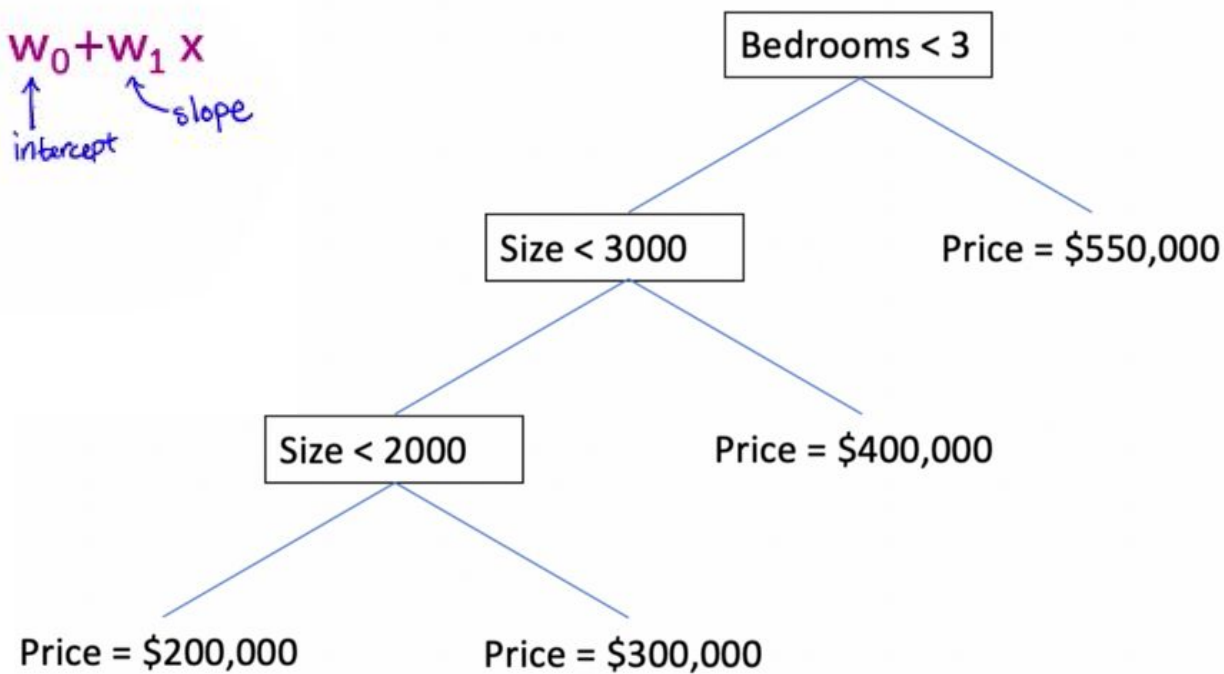
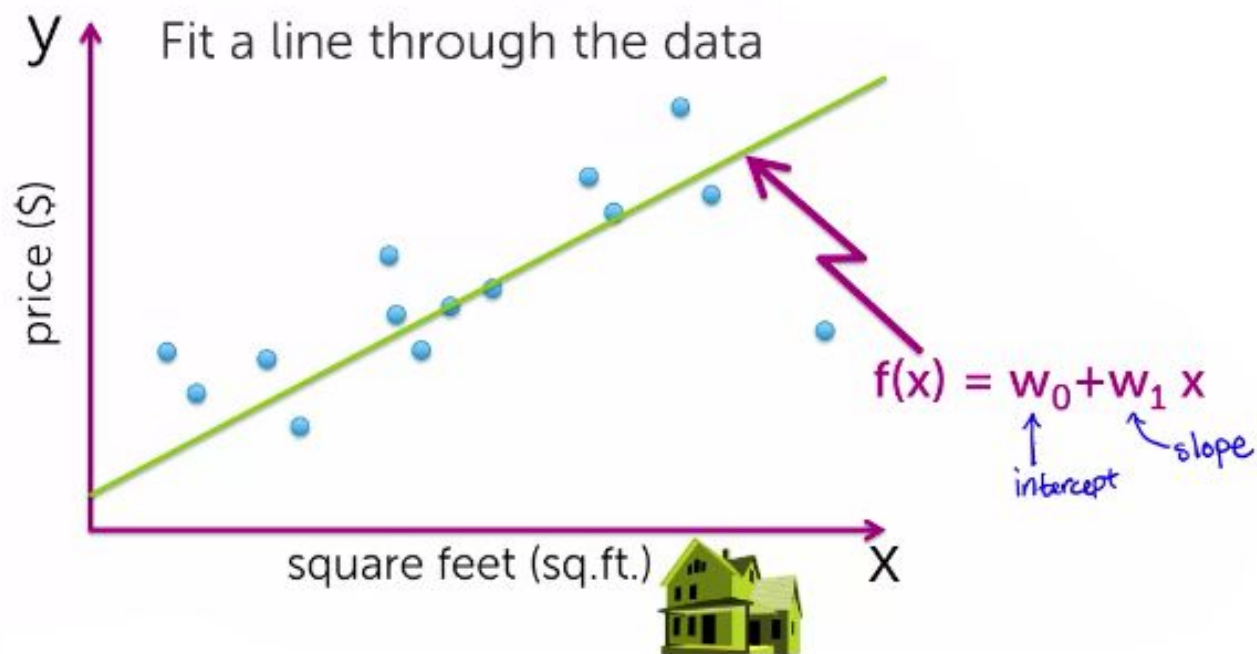
# СТРУКТУРА КУРСА



## В ЭТОМ УРОКЕ

- Библиотека Scikit-learn
- Подготовка данных для последующего моделирования
- Построение предиктивных моделей в Python
- Использование ключевых метрик качества моделей

# Модель машинного обучения



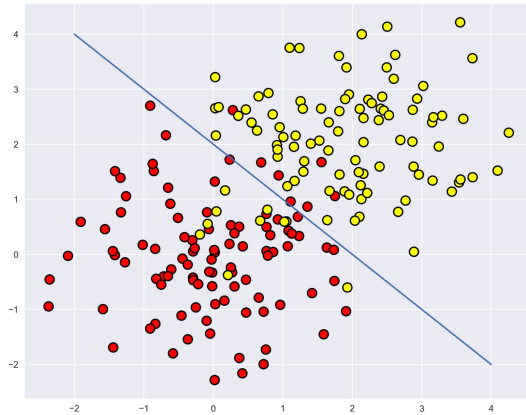
# Библиотека Scikit-Learn

программная библиотека на языке Python для машинного обучения с открытым исходным кодом, с помощью которой можно реализовать алгоритмы:

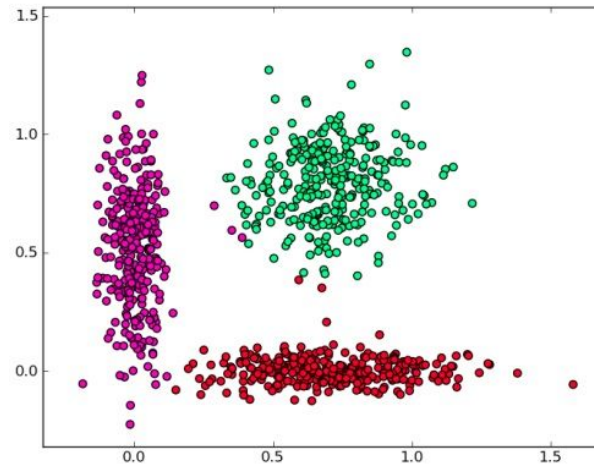
- классификации
- регрессии
- кластеризации



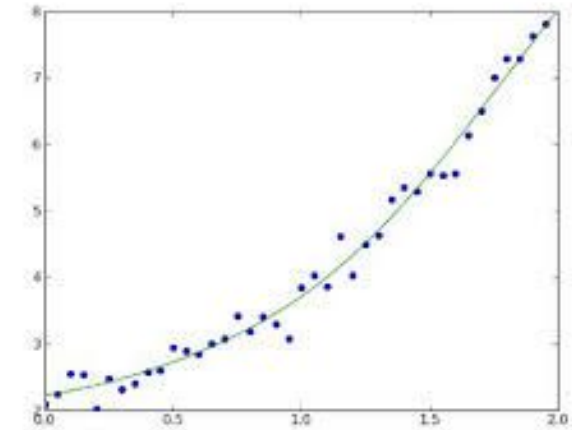
# Основные типы задач



Классификация  
(обучение с учителем)



Кластеризация  
(обучение без учителя)



Регрессия  
(обучение с учителем)

# Примеры бизнес-задач

## Классификация

- Кредитный скоринг
- Распределение заявок helpdesk

## Кластеризация

- Маркетинговые исследования
- Выявление аномалий в операциях

## Регрессия

- Оценка стоимости недвижимости
- Прогнозирование продаж

# Процесс





# Подготовка данных. Toy datasets.

Набор данных	Описание	Размер	Использование
load_boston()	Данные о ценах на недвижимость в Бостоне	506	регрессия
load_iris()	Данные измерений четырех параметров цветков ириса	150	классификация
load_digits()	Данные изображений цифр от 0 до 9	5620	классификация
load_wine()	Данные по химической классификации вин	178	классификация

```
[3] boston_data = load_boston()
    print(boston_data['DESCR'])

.. _boston_dataset:

Boston house prices dataset
-----

**Data Set Characteristics:**

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

:Attribute Information (in order):
- CRIM      per capita crime rate by town
- ZN        proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS     proportion of non-retail business acres per town
- CHAS      Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
```

# Подготовка данных. Замена незаполненных значений на NaN.

```
[45] test_data = pd.read_csv('TestForClean.csv', sep=';')
test_data.tail()
```

	Price	IsNew	Color	City	Year
95	15.0	Yes	Yellow	NaN	1981
96	46.0	No	Orange	NaN	unknown
97	39.0	Yes	Red	NaN	[1965]
98	32.0	Yes	Blue	London	[1971]
99	32.0	Yes	Blue	London	[1964]

```
test_data['Year'].replace("unknown", np.NaN, inplace = True)
test_data.tail()
```

	Price	IsNew	Color	City	Year
95	15.0	Yes	Yellow	NaN	1981
96	46.0	No	Orange	NaN	NaN
97	39.0	Yes	Red	NaN	[1965]
98	32.0	Yes	Blue	London	[1971]
99	32.0	Yes	Blue	London	[1964]

# Подготовка данных. Приведение колонок с числовыми данными к числовому типу с помощью RegExpr.

	Price	IsNew	Color	City	Year
95	15.0	Yes	Yellow	NaN	1981
96	46.0	No	Orange	NaN	NaN
97	39.0	Yes	Red	NaN	[1965]
98	32.0	Yes	Blue	London	[1971]
99	32.0	Yes	Blue	London	[1964]

Уберем [] из колонки Year

```
test_data['Year'] = test_data['Year'].str.replace(r"^[0-9]", "")  
test_data.tail()
```

	Price	IsNew	Color	City	Year
95	15.0	Yes	Yellow	NaN	1981
96	46.0	No	Orange	NaN	NaN
97	39.0	Yes	Red	NaN	1965
98	32.0	Yes	Blue	London	1971
99	32.0	Yes	Blue	London	1964

## Подготовка данных. Приведение колонок с числовыми данными к числовому типу.

```
[80] test_data.dtypes
```

```
Price    float64
IsNew     object
Color     object
City      object
Year      object
dtype: object
```

```
[81] test_data["Year"] = test_data["Year"].fillna(0).astype('int16')
test_data.dtypes
```

```
Price    float64
IsNew     object
Color     object
City      object
Year      int16
dtype: object
```

# Подготовка данных. Удаление колонок с высоким процентом содержания пустых значений.

Удалим колонки с содержанием пустых значений более 50%

```
[110] for column in test_data.columns:  
      print("{}: {}".format(column, round(np.mean(test_data[column].isnull()*100)))
```

```
Price: 5.0%  
IsNew: 0.0%  
Color: 0.0%  
City: 82.0%  
Year: 0.0%
```



```
test_data.drop('City', axis = 1).head()
```

	Price	IsNew	Color	Year
0	44.0	Yes	Red	1990
1	35.0	No	Blue	1954
2	50.0	No	Yellow	1977
3	12.0	Yes	Orange	1963
4	38.0	No	Red	1987

# Подготовка данных. Удаление аномальных значений.

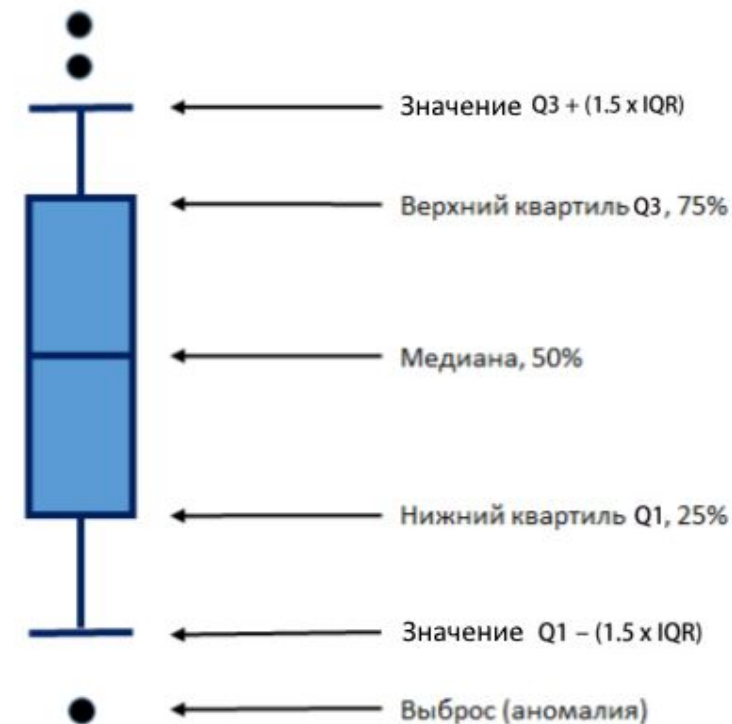
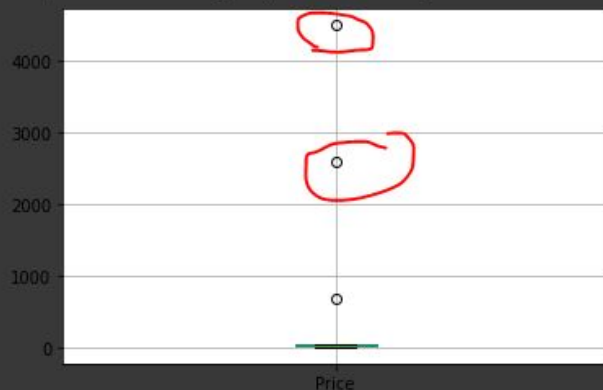
```
[27] test_data.Price.hist(bins=10)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3b4e85a978>



```
[22] test_data.boxplot(column=['Price'])
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3b4eb75278>



## Outliers Formula

Lower Outlier =  $Q1 - (1.5 \times IQR)$

Higher Outlier =  $Q3 + (1.5 \times IQR)$

## Подготовка данных. Заполняем отсутствующие значения.

```
[151] test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 100 entries, 0 to 99  
Data columns (total 4 columns):  
#   Column  Non-Null Count  Dtype  
---  ---  
0   Price   95 non-null     float64  
1   IsNew    100 non-null    object  
2   Color    100 non-null    object  
3   Year     100 non-null    int16  
dtypes: float64(1), int16(1), object(2)  
memory usage: 2.7+ KB
```

```
[154] test_data.Price = test_data.Price.fillna(test_data.Price.median())  
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 100 entries, 0 to 99  
Data columns (total 4 columns):  
#   Column  Non-Null Count  Dtype  
---  ---  
0   Price   100 non-null    float64  
1   IsNew    100 non-null    object  
2   Color    100 non-null    object  
3   Year     100 non-null    int16  
dtypes: float64(1), int16(1), object(2)  
memory usage: 2.7+ KB
```



# Подготовка данных. Масштабирование признаков.

```
[117] # Статистика для числовых столбцов  
x_train.describe()
```

	LSTAT	RM
count	404.000000	404.000000
mean	12.478045	6.288455
std	7.038575	0.692931
min	1.730000	3.561000
25%	6.840000	5.888000
50%	11.160000	6.209000
75%	16.992500	6.629250
max	37.970000	8.780000



```
from sklearn.preprocessing import Normalizer, StandardScaler
```

```
#transformer = StandardScaler().fit(x_train.loc[:, x_train.columns])
```

```
transformer = Normalizer().fit(x_train.loc[:, x_train.columns])
```

```
x_train = pd.DataFrame(transformer.transform(x_train.loc[:, x_train.columns]), index = x_train.index, columns = x_train.columns)
```

```
x_test = pd.DataFrame(transformer.transform(x_test.loc[:, x_test.columns]), index = x_test.index, columns = x_test.columns)
```

```
x_train.describe()
```

	LSTAT	RM
count	404.000000	404.000000
mean	0.812382	0.515042
std	0.168945	0.215430
min	0.225078	0.108339
25%	0.727078	0.338038
50%	0.874458	0.485099
75%	0.941132	0.686554
max	0.994114	0.974341



# Подготовка данных. Обработка категориальных признаков.

```
[193] test_data.nunique()
```

```
Price    34  
IsNew     2  
Color     4  
Year    38  
dtype: int64
```

```
[196] is_new_map = {'Yes': 1, 'No': 0}  
test_data.IsNew.map(is_new_map).head()
```

```
0    1  
1    0  
2    0  
3    1  
4    0  
Name: IsNew, dtype: int64
```



```
pd.get_dummies(test_data.Color, prefix='Color_').head()
```

	Color_Blue	Color_Orange	Color_Red	Color_Yellow
0	0	0	1	0
1	1	0	0	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	0

# Обучение модели. Выбор признаков: матрица корреляции.

Посмотрим матрицу корреляции

```
[73] x_corr = x.copy()
     x_corr['MEDV'] = y
     correlation_matrix = x_corr.corr().round(2)
     plt.subplots(figsize=(10,7))
     sns.heatmap(data=correlation_matrix, annot=True)
```

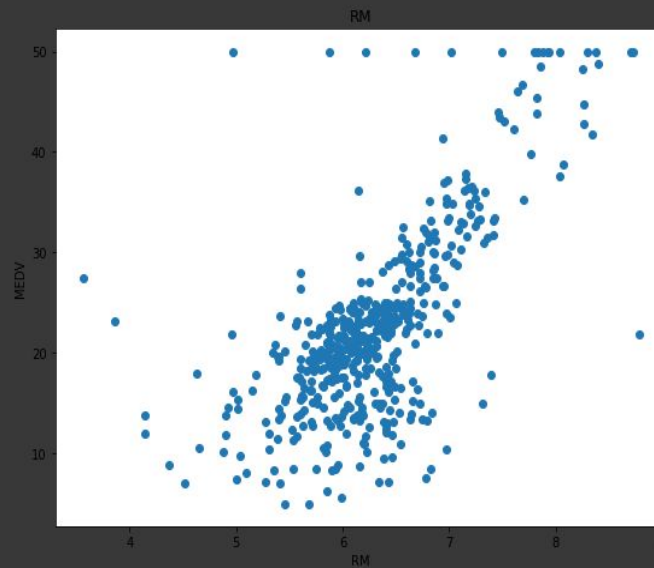
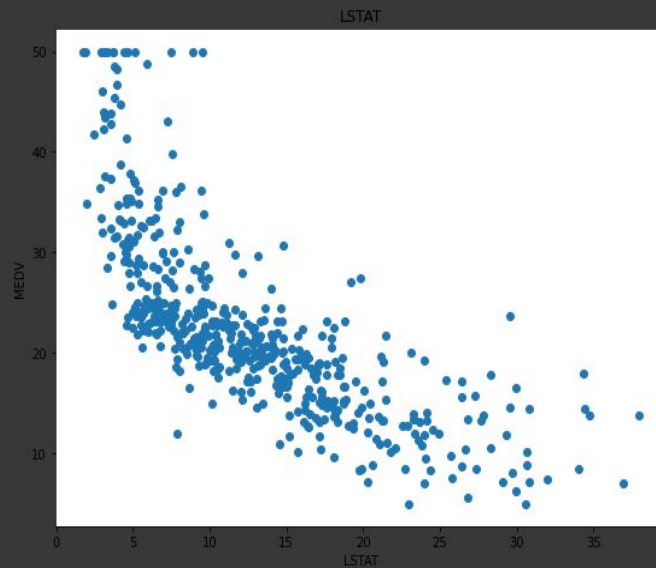
<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb565d8ab00>



# Обучение модели. Анализ выбранных признаков.

Посмотрим на LSTAT и RM

```
[71] plt.figure(figsize=(20, 7))  
  
features = ['LSTAT', 'RM']  
  
for i, col in enumerate(features):  
    plt.subplot(1, len(features), i+1)  
    plt.scatter(x[col], y, marker='o')  
    plt.title(col)  
    plt.xlabel(col)  
    plt.ylabel('MEDV')
```



# Обучение модели. Разделение на обучающую и тестовую выборки.

Загрузим данные о признаках в "x", а целевые значения в "y"

```
[10] x = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
     y = boston_data.target
```

Оставим только два признака LSTAT и RM

```
[ ] x_scaled = x_scaled[['LSTAT', 'RM']]
    x_scaled.shape

(506, 2)
```

Разделим данные на обучающую и тестовую выборки в пропорции 80 и 20

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.2, random_state=5)
```

# Обучение модели. Сохранение и загрузка.

Метод линейной регрессии

```
[52] from sklearn.linear_model import LinearRegression  
     regr = LinearRegression()  
     regr.fit(x_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Сохраняем обученную модель на диск

```
[36] import pickle  
     pickle.dump(regr, open('boston_train_model.sav', 'wb'))
```

Закгружаем сохраненную ранее модель

```
[38] regr = joblib.load('boston_train_model.sav')
```



# Предсказание результата. Оценка качества регрессии.

```
[78] from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
pred_train = regr.predict(x_train)
pred_test = regr.predict(x_test)
print("Средняя абсолютная ошибка (MAE) модели на обучающей выборке {}".format(mean_absolute_error(y_train, pred_train)))
print("Средняя абсолютная ошибка (MAE) модели на тестовой выборке {}".format(mean_absolute_error(y_test, pred_test)))
print("Средняя квадратическая ошибка (MSE) модели на обучающей выборке {}".format(mean_squared_error(y_train, pred_train)))
print("Средняя квадратическая ошибка (MSE) модели на тестовой выборке {}".format(mean_squared_error(y_test, pred_test)))
print("Коэффициент детерминации (R^2) модели на обучающей выборке {}".format(r2_score(y_train, pred_train)))
print("Коэффициент детерминации (R^2) модели на тестовой выборке {}".format(r2_score(y_test, pred_test)))
```

```
Средняя абсолютная ошибка (MAE) модели на обучающей выборке 3.3500095196484523
Средняя абсолютная ошибка (MAE) модели на тестовой выборке 3.2132704958423783
Средняя квадратическая ошибка (MSE) модели на обучающей выборке 22.477090408387635
Средняя квадратическая ошибка (MSE) модели на тестовой выборке 20.869292183770735
Коэффициент детерминации (R^2) модели на обучающей выборке 0.7383393920590519
Коэффициент детерминации (R^2) модели на тестовой выборке 0.7334492147453086
```

# Предсказание результата. Оценка качества классификации.

## Матрица ошибок (confusion matrix)

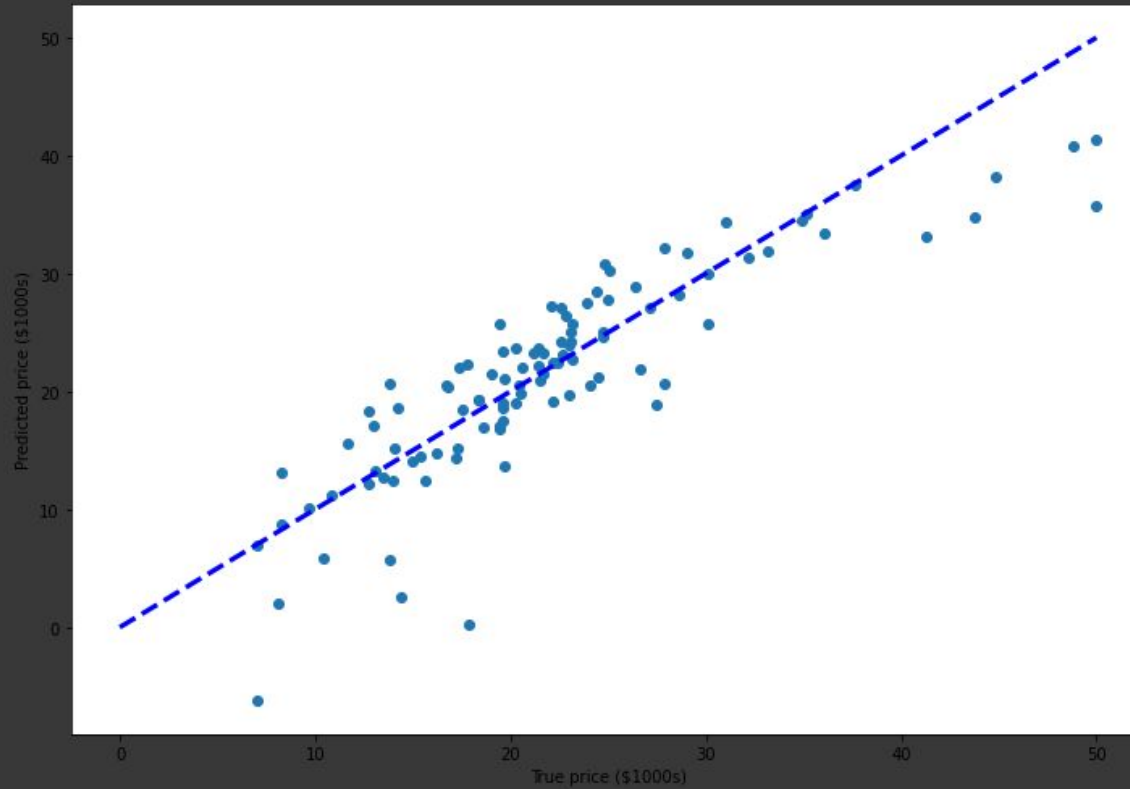
	Actual positive	Actual negative
Predicted positive	True positive	False positive
Predicted negative	False negative	True negative

Основные метрики оценки качества классификации:

- Accuracy - процент правильных предсказаний  $(TP + TN) / (TP + TN + FP + FN)$
- Precision (точность) - процент предсказанных как положительные и действительно являющимися положительными  $TP / (TP + FP)$
- Recall (полнота) - процент правильно определенных как положительные из общего числа истинных положительных  $TP / (TP + FN)$

# Оценка качества предсказания. Оценка текущего предсказания.

```
[79] plt.figure(figsize=(10, 7))  
plt.scatter(y_test, pred_test)  
plt.plot([0, 50], [0, 50], '--k', color='blue', linewidth=3)  
plt.xlabel('True price ($1000s)')  
plt.ylabel('Predicted price ($1000s)')  
plt.tight_layout()
```





# Оценка качества предсказания. Оценка альтернативной модели.

## Метод "Градиентный boosting"

```
x = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
x_scaled = pd.DataFrame(preprocessing.normalize(x, axis = 0, norm = 'max'), columns = x.columns)
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.2, random_state=5)

from sklearn.ensemble import GradientBoostingRegressor
regr = GradientBoostingRegressor()
regr.fit(x_train, y_train)

pred_train = regr.predict(x_train)
pred_test = regr.predict(x_test)
print("Средняя абсолютная ошибка (MAE) модели на обучающей выборке {}".format(mean_absolute_error(y_train, pred_train)))
print("Средняя абсолютная ошибка (MAE) модели на тестовой выборке {}".format(mean_absolute_error(y_test, pred_test)))
print("Средняя квадратическая ошибка (MSE) модели на обучающей выборке {}".format(mean_squared_error(y_train, pred_train)))
print("Средняя квадратическая ошибка (MSE) модели на тестовой выборке {}".format(mean_squared_error(y_test, pred_test)))
print("Коэффициент детерминации (R^2) модели на обучающей выборке {}".format(r2_score(y_train, pred_train)))
print("Коэффициент детерминации (R^2) модели на тестовой выборке {}".format(r2_score(y_test, pred_test)))
```

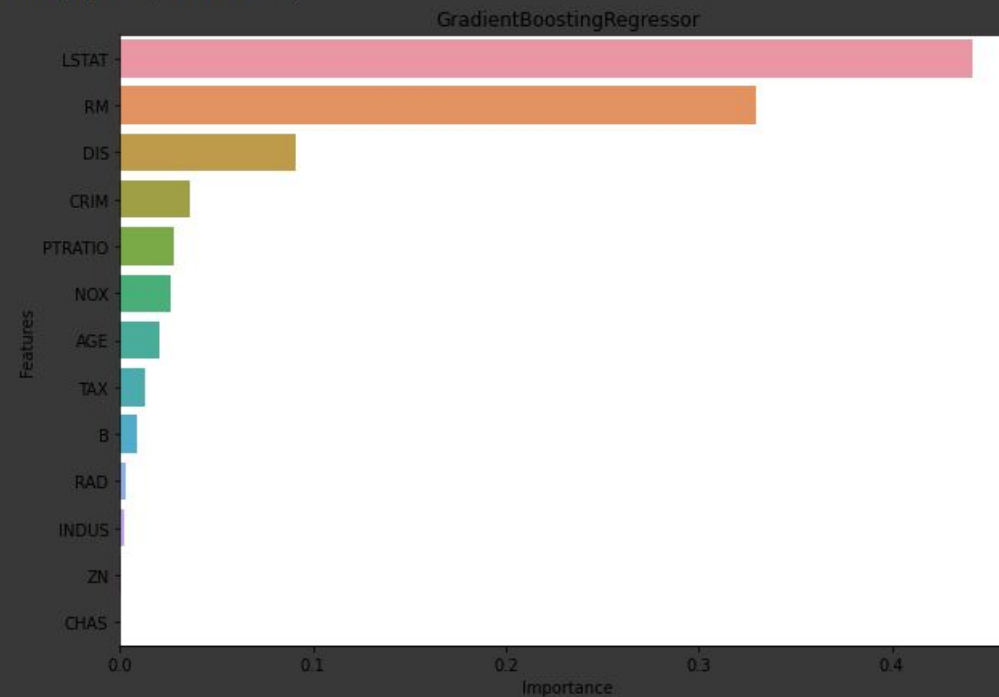
```
Средняя абсолютная ошибка (MAE) модели на обучающей выборке 1.09514690693163
Средняя абсолютная ошибка (MAE) модели на тестовой выборке 1.9691033158663265
Средняя квадратическая ошибка (MSE) модели на обучающей выборке 1.8932814816592998
Средняя квадратическая ошибка (MSE) модели на тестовой выборке 8.931110956145188
Коэффициент детерминации (R^2) модели на обучающей выборке 0.9779599060868908
Коэффициент детерминации (R^2) модели на тестовой выборке 0.8859283478522296
```

# Оценка качества предсказания. Анализ важности признаков.

## Анализ важности признаков

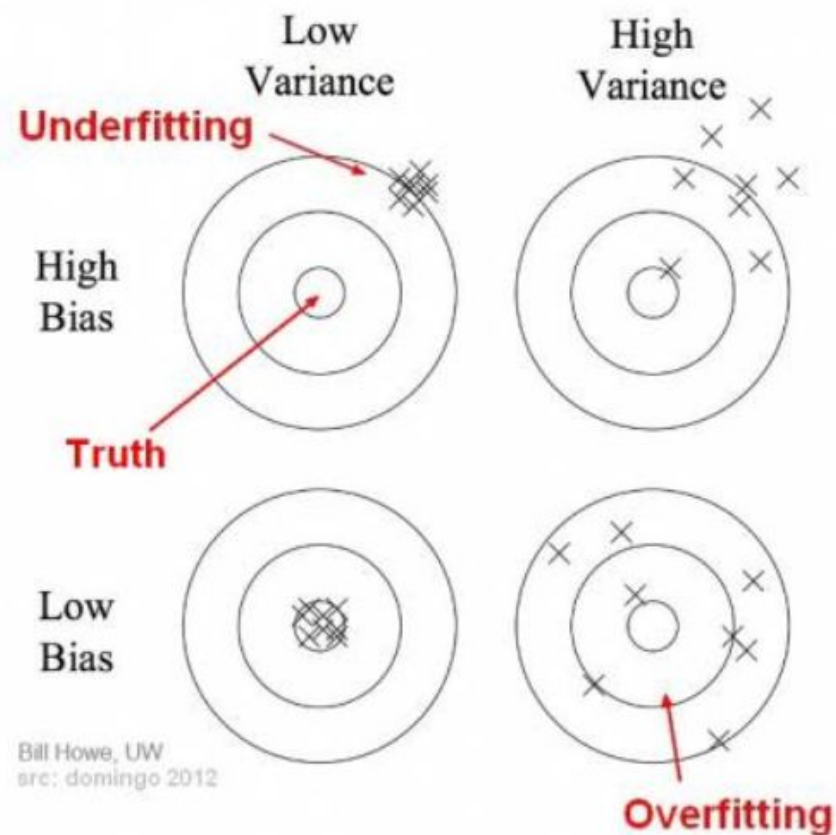
```
feature_importance = pd.DataFrame({'feature': list(x.columns), 'importance': regr.feature_importances_}).sort_values('importance', ascending = False).reset_index(drop = True)
plt.figure(figsize = (10, 7))
sns.barplot(x = feature_importance['importance'], y = feature_importance['feature'])
plt.title('GradientBoostingRegressor')
plt.xlabel('Importance')
plt.ylabel('Features')
```

Text(0, 0.5, 'Features')

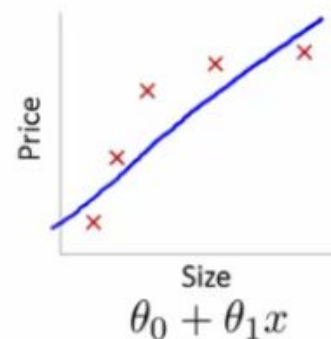


# Оценка качества предсказания

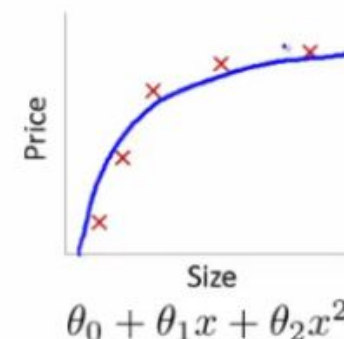
## Недообучение и переобучение модели



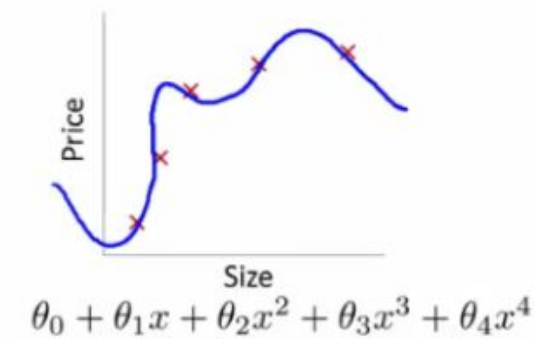
$$\text{prediction\_error}(X) = \text{noise}(X) + \text{bias}(X) + \text{variance}(X)$$



High bias  
(underfit)



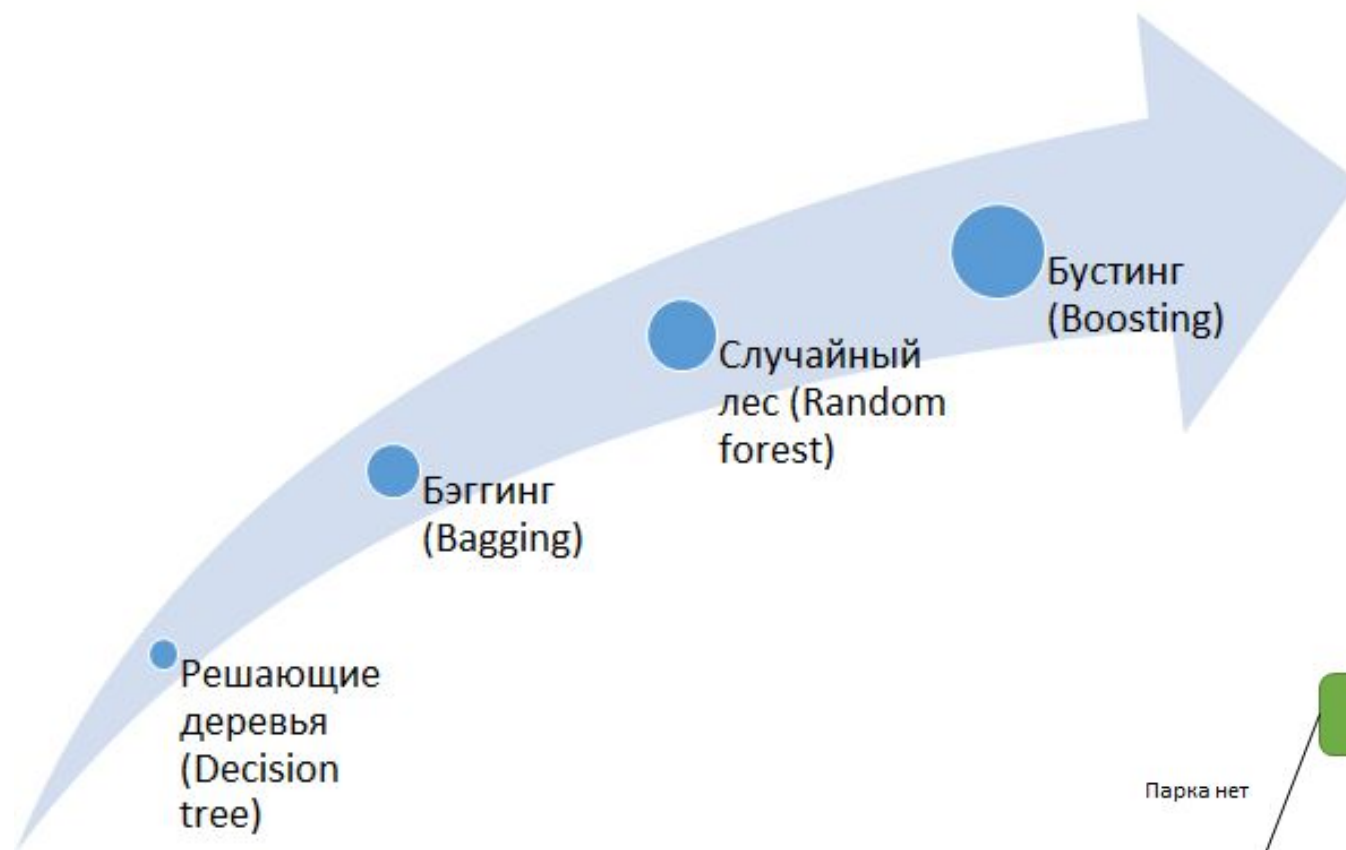
"Just right"



High variance  
(overfit)

# Оценка качества предсказания

## Решающие деревья





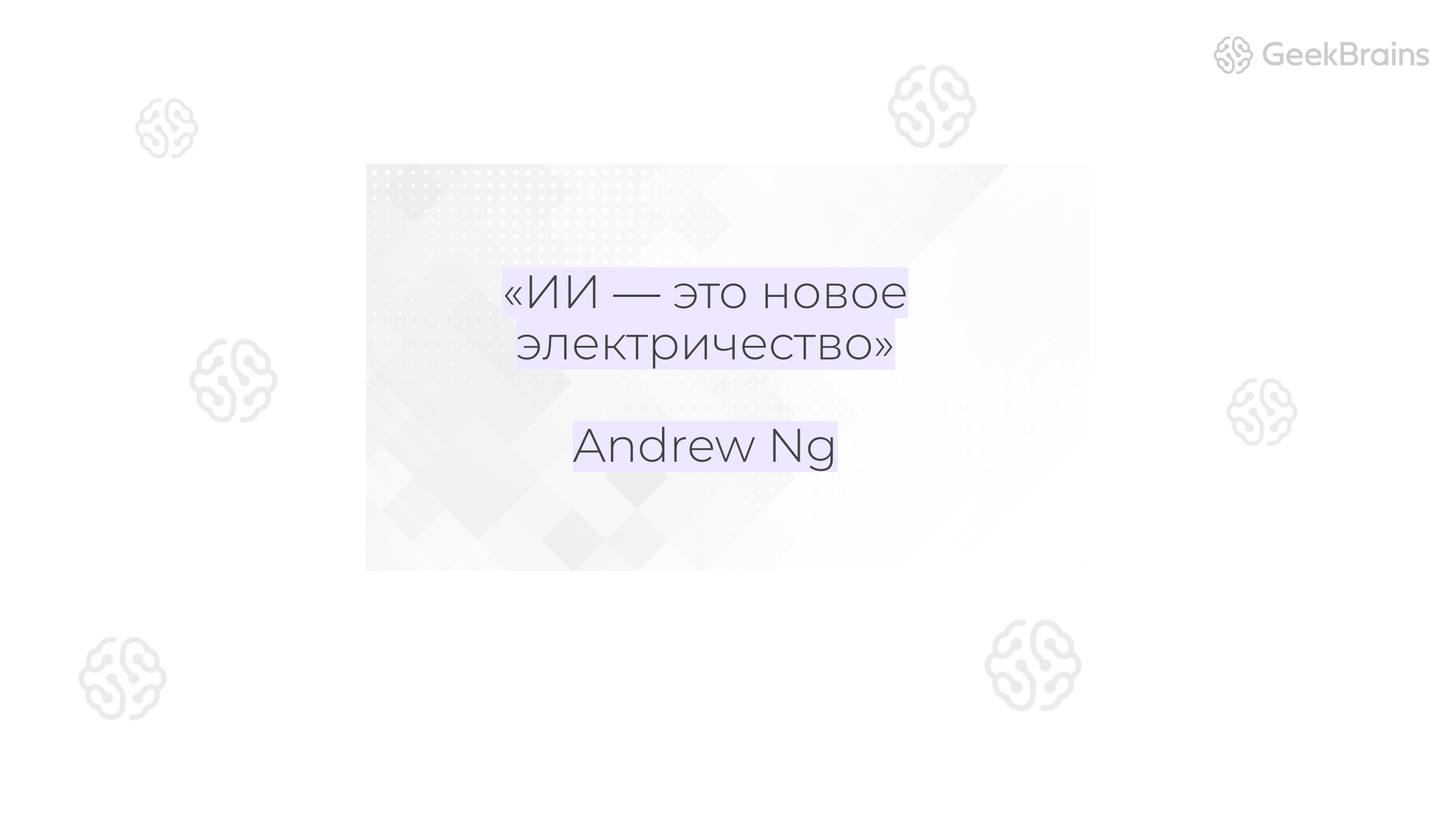
# Оценка качества предсказания

## Гиперпараметрическая оптимизация модели

### Гиперпараметрическая оптимизация модели

```
regr = GradientBoostingRegressor(  
    loss = 'lad',  
    max_depth = 3,  
    max_features = 'auto',  
    min_samples_leaf = 6,  
    min_samples_split = 8,  
    n_estimators = 1100,  
)  
regr.fit(x_train, y_train)  
  
pred_train = regr.predict(x_train)  
pred_test = regr.predict(x_test)  
print("Средняя абсолютная ошибка (MAE) модели на обучающей выборке {}".format(mean_absolute_error(y_train, pred_train)))  
print("Средняя абсолютная ошибка (MAE) модели на тестовой выборке {}".format(mean_absolute_error(y_test, pred_test)))  
print("Средняя квадратическая ошибка (MSE) модели на обучающей выборке {}".format(mean_squared_error(y_train, pred_train)))  
print("Средняя квадратическая ошибка (MSE) модели на тестовой выборке {}".format(mean_squared_error(y_test, pred_test)))  
print("Коэффициент детерминации (R^2) модели на обучающей выборке {}".format(r2_score(y_train, pred_train)))  
print("Коэффициент детерминации (R^2) модели на тестовой выборке {}".format(r2_score(y_test, pred_test)))
```

Средняя абсолютная ошибка (MAE) модели на обучающей выборке 0.7533474704686788  
Средняя абсолютная ошибка (MAE) модели на тестовой выборке 1.9598292877422614  
Средняя квадратическая ошибка (MSE) модели на обучающей выборке 2.6117203461167073  
Средняя квадратическая ошибка (MSE) модели на тестовой выборке 7.731045790044177  
Коэффициент детерминации (R^2) модели на обучающей выборке 0.9695964058906118  
Коэффициент детерминации (R^2) модели на тестовой выборке 0.9012560508506947



«ИИ — это новое  
электричество»

Andrew Ng



## РЕЗЮМЕ УРОКА

- Научились использовать «Toy datasets»
- Узнали о том, как готовить обучающую и тестовую выборки
- Научились обучать модель и анализировать точность предсказания
- Смогли визуально сравнить точность предсказания



## ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Взять все признаки из датасета с урока. Применить модель Random Forest.
2. Рассчитайте MAE, MSE, R2 на тренировочной и тестовой выборке.
3. Посмотрите, как показатели качества соотносятся с коэффициентом корреляции этих признаков и целевого значения, а также между собой.
4. Сравнить показатели качества новой модели с моделью с урока и сделать вывод.

Результат - ссылка [на готовый ноутбук в Colab](#).





# Дополнительные материалы

1. Руководство пользователя



# ИТОГОВАЯ РАБОТА (для ознакомления)

1. Скачать данные о рынке недвижимости с открытого соревнования Сбербанка по машинному обучению <https://www.kaggle.com/c/sberbank-russian-housing-market/data>. Скачиваем только файлы train.csv.zip, macro.csv.zip и сливаем в один датафрейм. Описание колонок - data\_dictionary.txt
  2. Провести чистку данных (пропуски данных, неинформативные признаки, аномалии, оцифровка текстовых столбцов и т.п.)
  3. Провести Exploratory Data Analysis. Проанализировать переменные, влияющие на зависимую переменную - стоимость недвижимости price\_doc.
  4. Разделить датасет на обучающие и тестовую выборки.
  5. Обучить регрессионную модель предсказывать стоимость недвижимости в зависимости от разных факторов и выявить показатели качества модели на обучающей и тестовой выборке.
  6. Реализовать методы feature engineering с целью повышения метрик качества модели.
- Результат - ссылка на готовый ноутбук в Colab с выводами.

ВАШИ ВОПРОСЫ