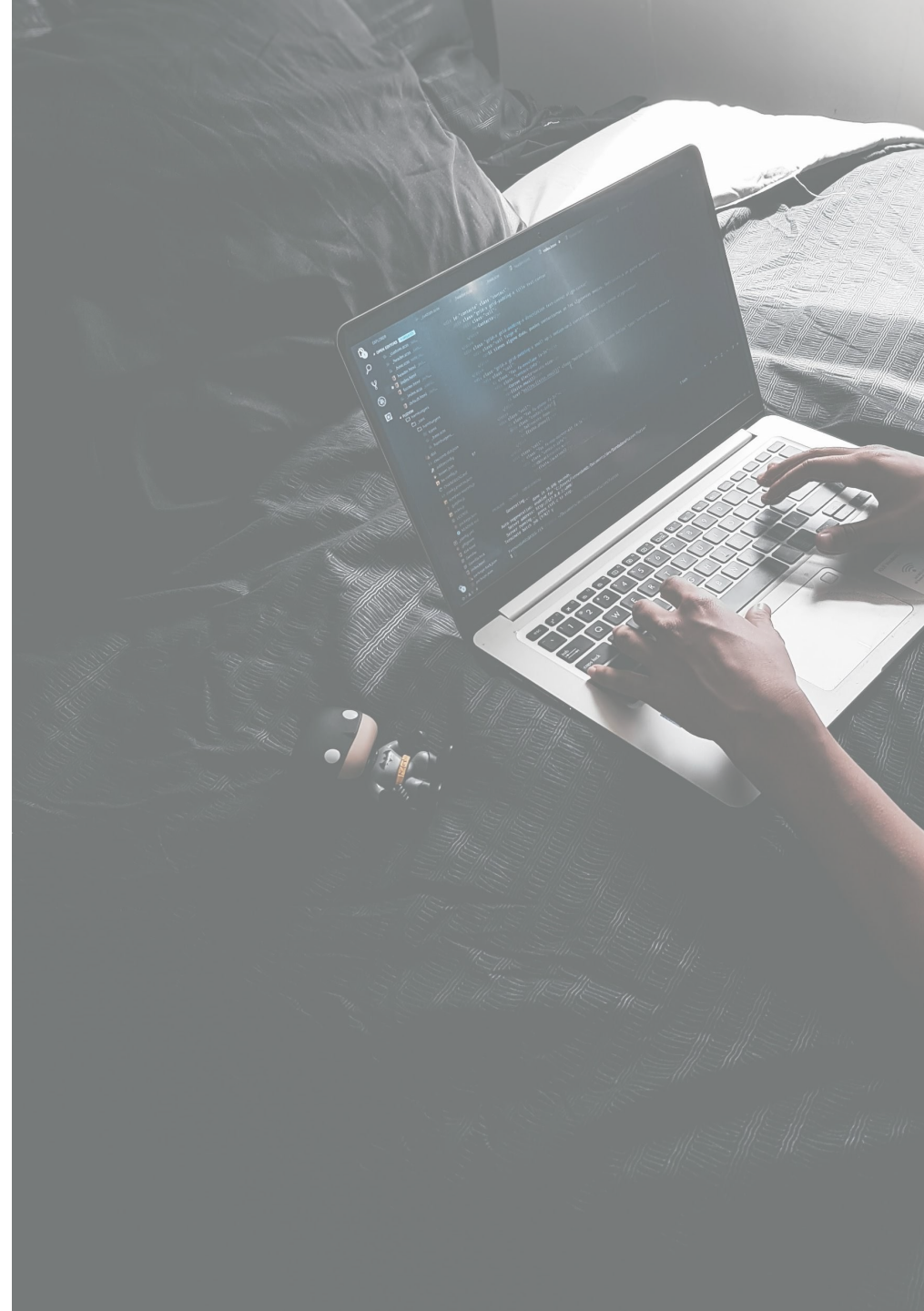
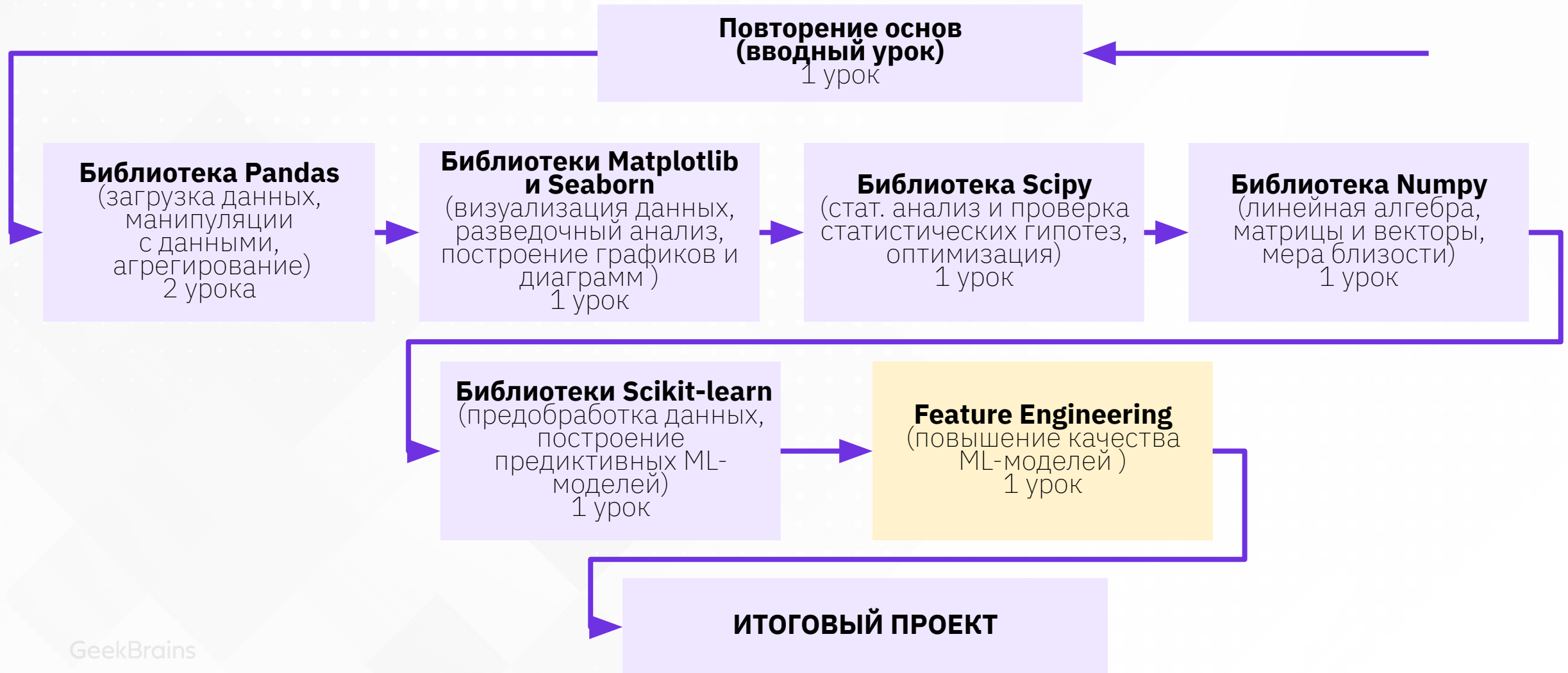


Повышение качества предиктивных моделей. Feature Engineering

Python для аналитиков / урок 8



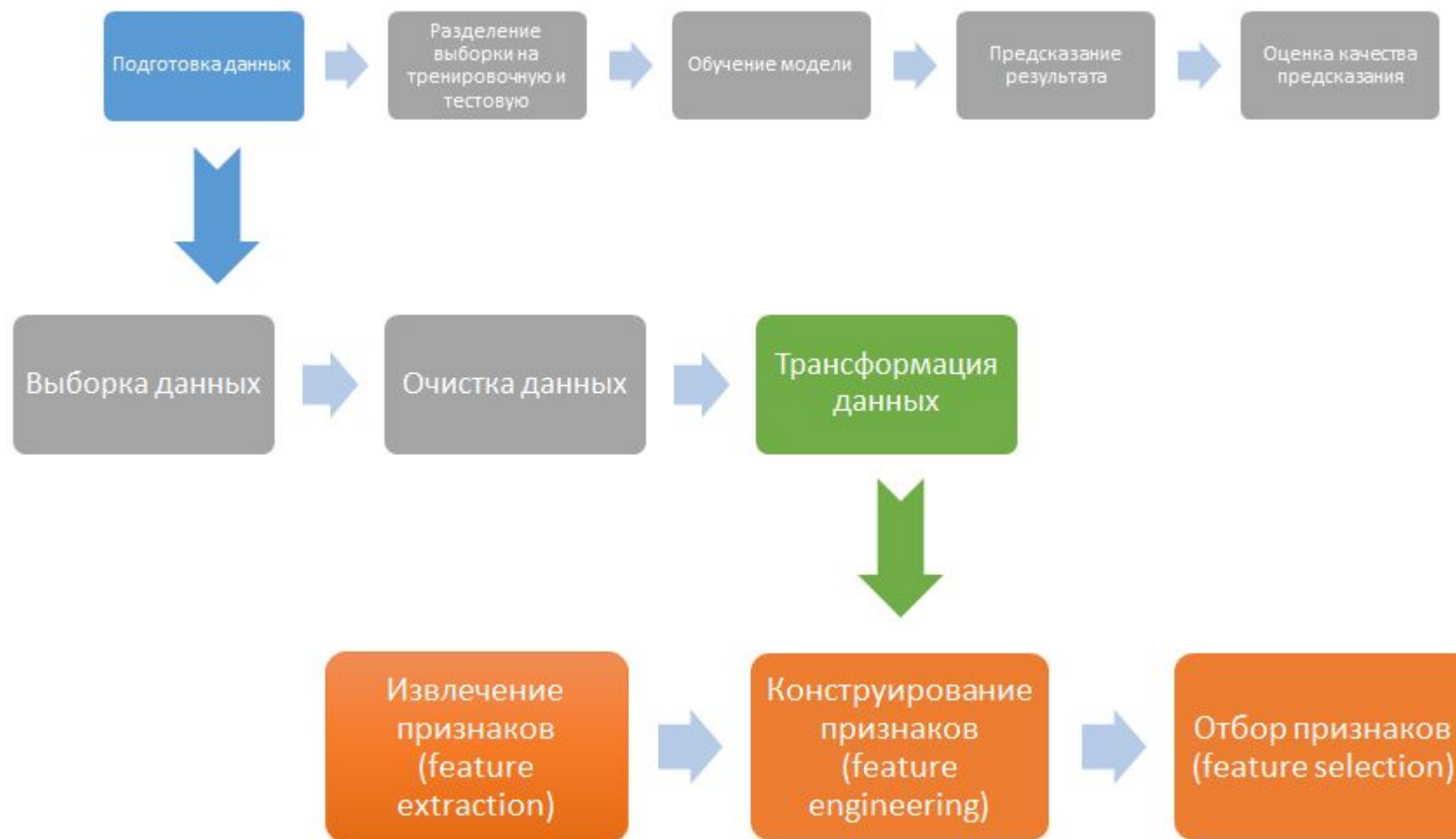
СТРУКТУРА КУРСА



В ЭТОМ УРОКЕ

- Факторы в модели и ограничения на их использование
- Kaggle и техники опытных кагглеров для повышения точности предсказательных моделей
- Разбор реализации основных техник генерации новых полезных фич
- Ключевые методы для отбора важных фич

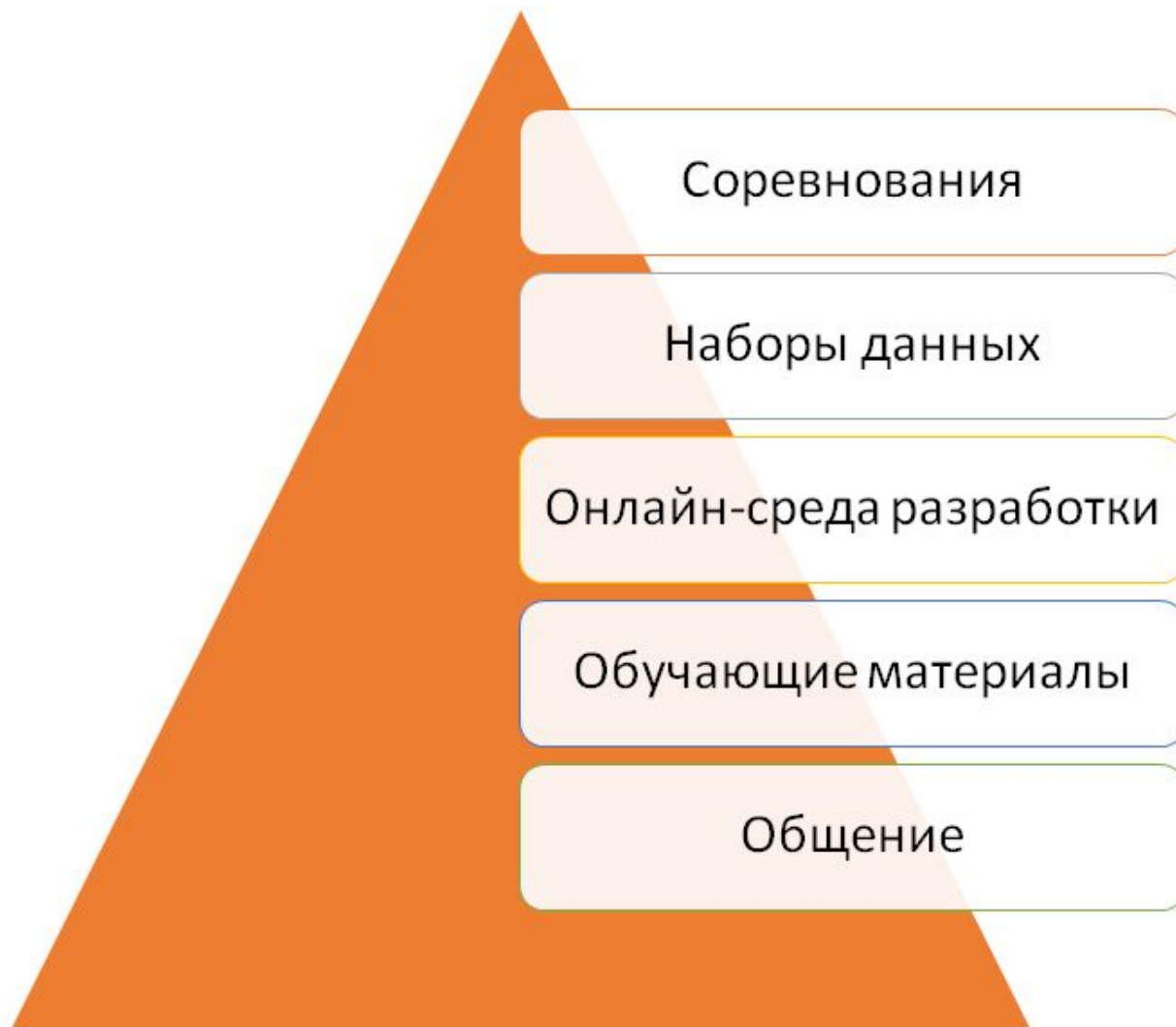
Feature Engineering



Классификация признаков



Kaggle



Извлечение признаков (feature extraction)



Тексты



Геоданные



Изображения



Дата и время

Конструирование признаков (feature engineering). Количественные

Нормализация и
стандартизация

Эффекты
взаимодействия

Заполнение пустых
значений

Бинаризация

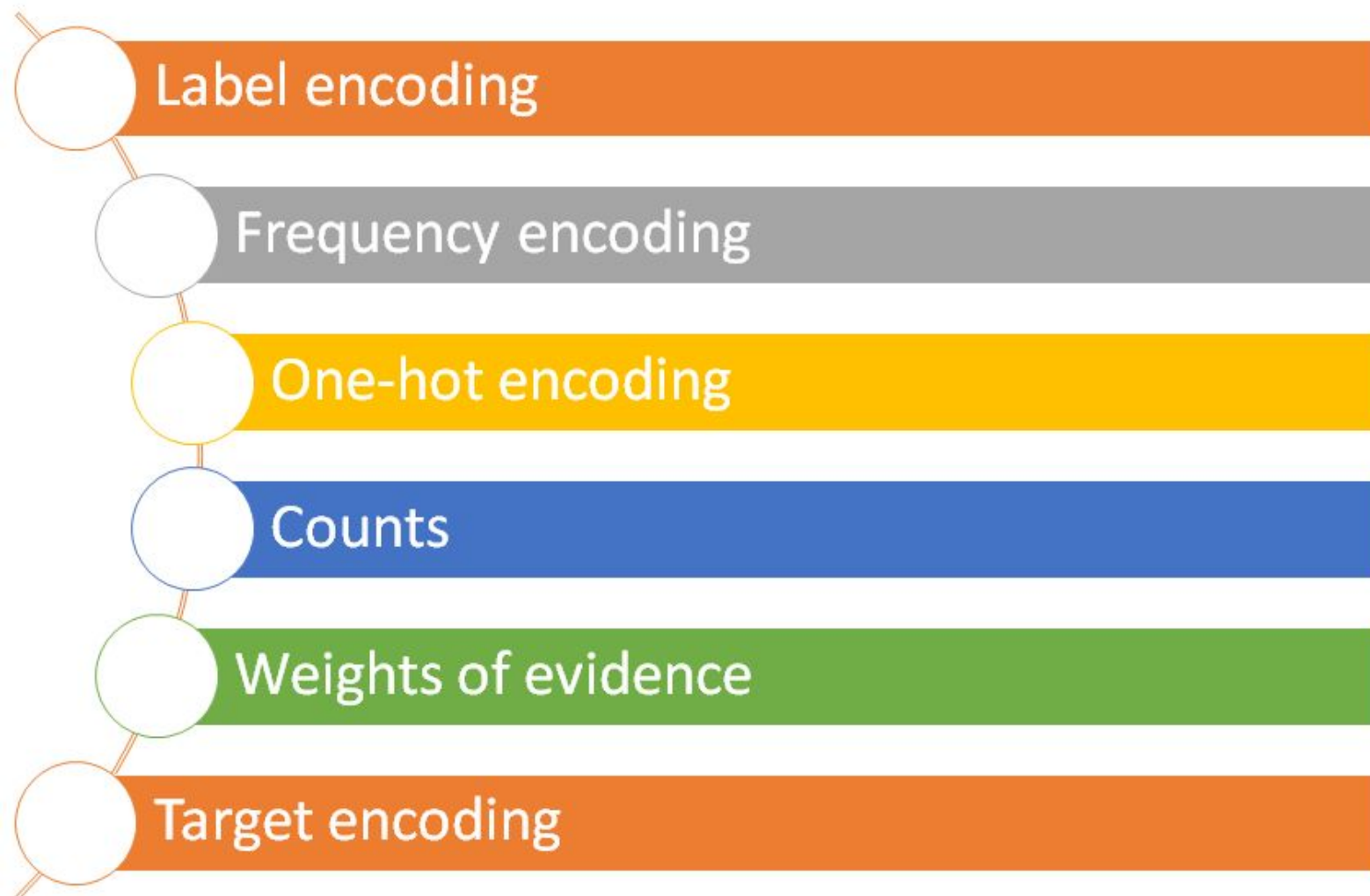
Биннинг

Логарифмирование

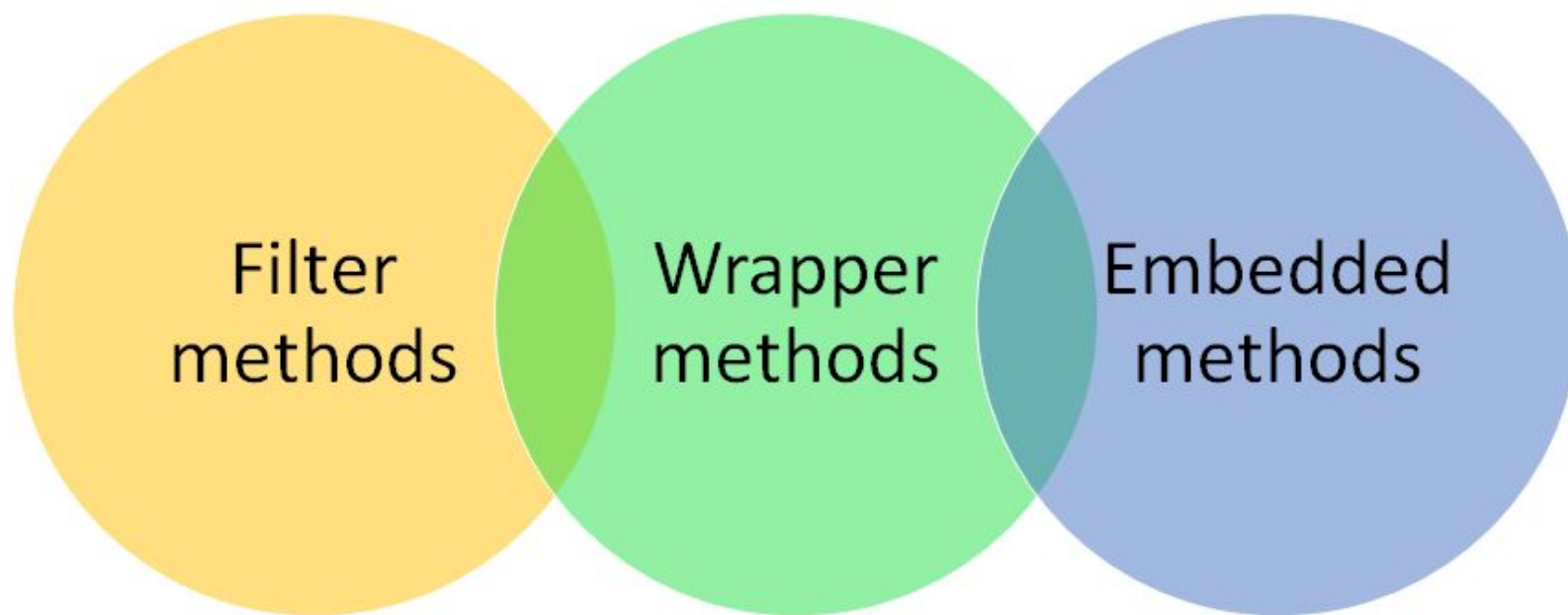
Винсоризация

Триминг

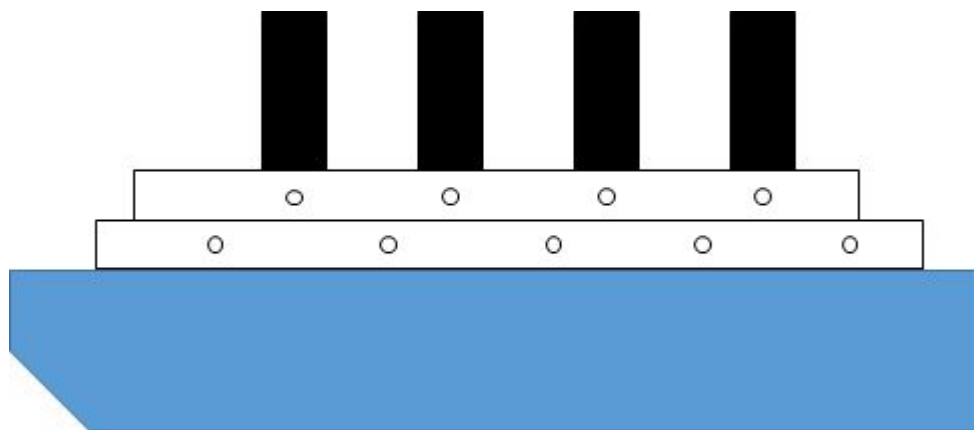
Конструирование признаков (feature engineering). Качественные



Отбор признаков (feature selection)



Практика. Выбираем dataset на Kaggle.



<https://www.kaggle.com/c/titanic/overview>

Практика. Загружаем dataset и смотрим описание.

```
x = pd.read_csv('titanic_train.csv', sep=',')
x.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

- PassengerId - Порядковый номер пассажира
- Survived - Выжил: 0 - нет, 1 - да (целевая переменная)
- Pclass - класс билета: 1-й (Upper), 2-й (Middle), 3-й (Lower)
- Name - Имя пассажира
- Sex - Пол пассажира
- Age - Возраст пассажира
- SibSp - Братьев/сестер/супругов на борту
- Parch - Детей/родителей на борту
- Ticket - Номер билета
- Fare - Стоимость билета
- Cabin - Номер каюты
- Embarked - Порт посадки: C = Cherbourg, Q = Queenstown, S = Southampton

Практика. Посмотрим какие признаки есть.

```
k.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 11 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   PassengerId  891 non-null    int64  
1   Pclass       891 non-null    int64  
2   Name         891 non-null    object  
3   Sex          891 non-null    object  
4   Age          714 non-null    float64  
5   SibSp        891 non-null    int64  
6   Parch        891 non-null    int64  
7   Ticket       891 non-null    object  
8   Fare         891 non-null    float64  
9   Cabin        204 non-null    object  
10  Embarked     889 non-null    object  
dtypes: float64(2), int64(4), object(5)  
memory usage: 76.7+ KB
```

Практика. Признак PassengerId.

- PassengerId - Порядковый номер пассажира

```
[32] x.PassengerId.describe()
```

```
count    891.000000
mean     446.000000
std      257.353842
min       1.000000
25%      223.500000
50%      446.000000
75%      668.500000
max      891.000000
Name: PassengerId, dtype: float64
```

```
[33] x.PassengerId.nunique()
```

```
891
```

```
[34] x.drop('PassengerId', axis = 'columns', inplace = True)
```

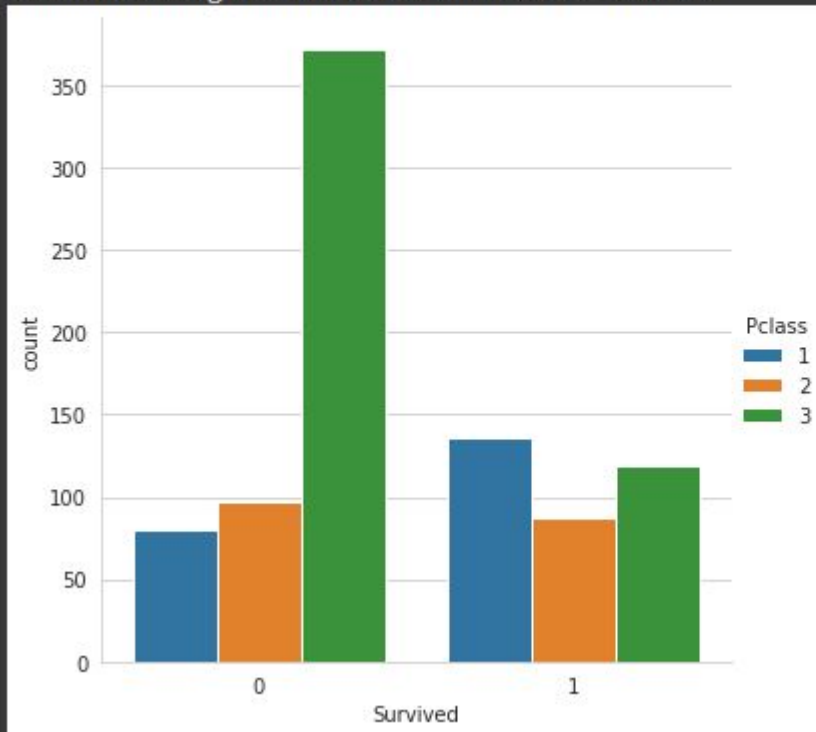
Практика. Признак Pclass.

- Pclass - класс билета: 1-й (Upper), 2-й (Middle), 3-й (Lower)



```
sns.set_style('whitegrid')  
sns.catplot(x = 'Survived', data = data, kind = 'count', hue = 'Pclass')
```

<seaborn.axisgrid.FacetGrid at 0x7f2a77f5d978>



Практика. Признак Name.

- Name - Имя пассажира

```
[89] x.Name.describe()
```

```
count      891
unique      891
top    Brown, Mr. Thomas William Solomon
freq              1
Name: Name, dtype: object
```

```
x.drop('Name', axis = 'columns', inplace = True)
```


Практика. Признак Sex.

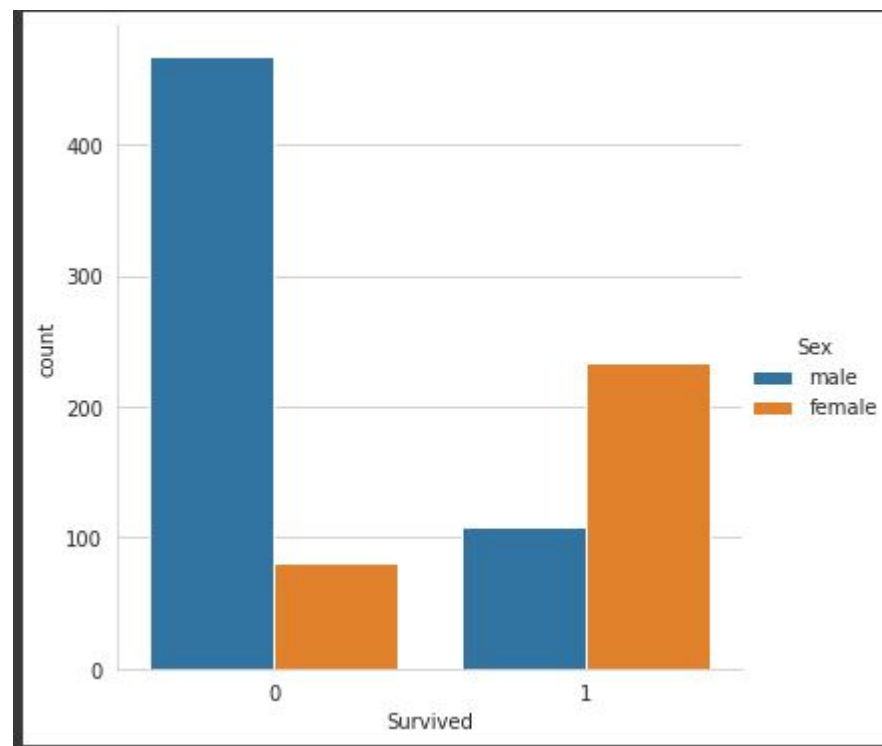
- Sex - Пол пассажира

```
[92] x.Sex.describe()
```

```
count      891  
unique        2  
top      male  
freq       577  
Name: Sex, dtype: object
```

```
labelEncoder = LabelEncoder()
```

```
x_train['Sex'] = labelEncoder.fit_transform(x_train['Sex'])  
x_test['Sex'] = labelEncoder.transform(x_test['Sex'])
```

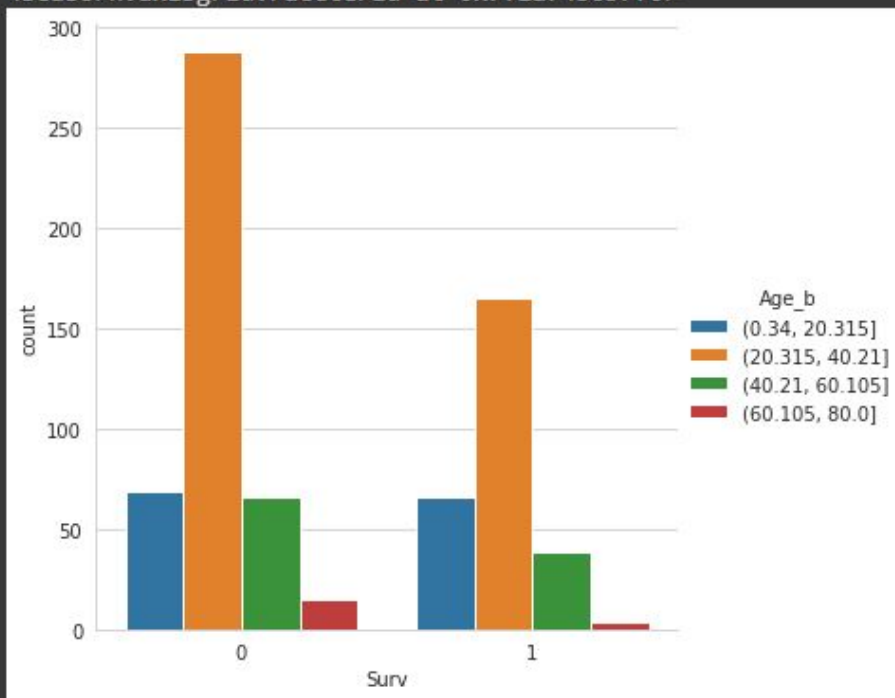


Практика. Признак Age.

- Age - Возраст пассажира

```
s, bins = pd.cut(x_train['Age'], 4, retbins=True)
x_train['Age_b'] = pd.cut(x_train['Age'], bins = bins, labels=False)
x_test['Age_b'] = pd.cut(x_test['Age'], bins = bins, labels=False)
x_corr = x_train.copy()
x_corr['Surv'] = y_train
sns.catplot(x = 'Surv', data = x_corr, kind = 'count', hue = 'Age_b')
```

<seaborn.axisgrid.FacetGrid at 0x7f2a745657f0>

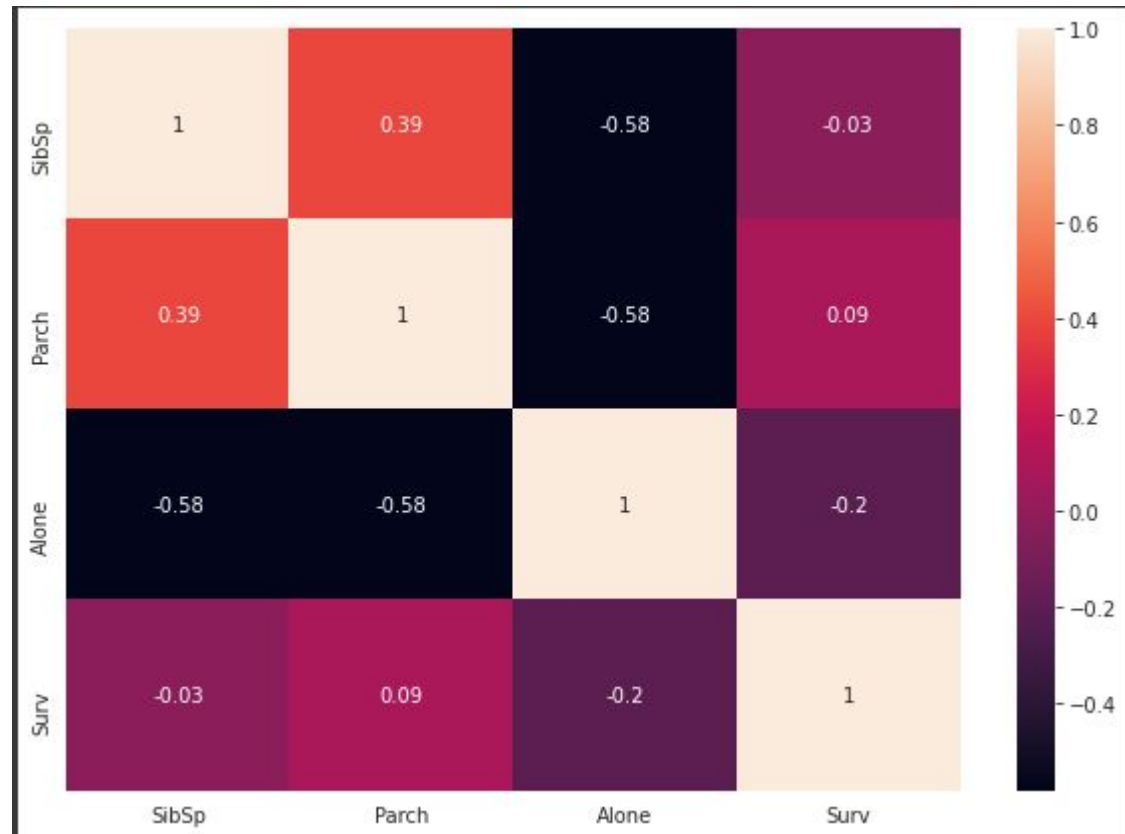


Практика. Признаки SibSp и Parch.

- SibSp - Братьев/сестер/супругов на борту
- Parch - Детей/родителей на борту

```
datasets = [x_train, x_test]
for ds in datasets:
    ds['Family_count'] = ds['SibSp'] + ds['Parch']
    ds.loc[ds['Family_count'] > 0, 'Alone'] = 0
    ds.loc[ds['Family_count'] == 0, 'Alone'] = 1

x_corr = x_train[['SibSp', 'Parch', 'Alone']].copy()
x_corr['Surv'] = y_train
correlation_matrix = x_corr.corr().round(2)
plt.subplots(figsize=(10,7))
sns.heatmap(data=correlation_matrix, annot=True)
```



Практика. Признак Ticket.

- Ticket - Номер билета

```
[294] x.Ticket.head(10)
```

```
0      A/5 21171
1      PC 17599
2  STON/O2. 3101282
3      113803
4      373450
5      330877
6      17463
7      349909
8      347742
9      237736
Name: Ticket, dtype: object
```

```
x.drop('Ticket', axis = 'columns', inplace = True)
```

Практика. Признак Fare.

- Fare - Стоимость билета

```

x_train['FarePerPassenger'] = x_train.apply(lambda row: row['Fare'] / row['Family_count'] if row['Alone'] == 0 else row['Fare'], axis = 1)
x_test['FarePerPassenger'] = x_test.apply(lambda row: row['Fare'] / row['Family_count'] if row['Alone'] == 0 else row['Fare'], axis = 1)

s, bins = pd.qcut(x_train['FarePerPassenger'], 6, retbins=True)
x_train['FarePerPassenger_b'] = pd.cut(x_train['FarePerPassenger'], bins = bins, labels=False)
x_test['FarePerPassenger_b'] = pd.cut(x_test['FarePerPassenger'], bins = bins, labels=False)

x_corr = x_train[['Fare', 'FarePerPassenger', 'FarePerPassenger_b']].copy()
x_corr['Surv'] = y_train
correlation_matrix = x_corr.corr().round(2)
plt.subplots(figsize=(10,7))
sns.heatmap(data=correlation_matrix, annot=True)

```



Практика. Признак Cabin.

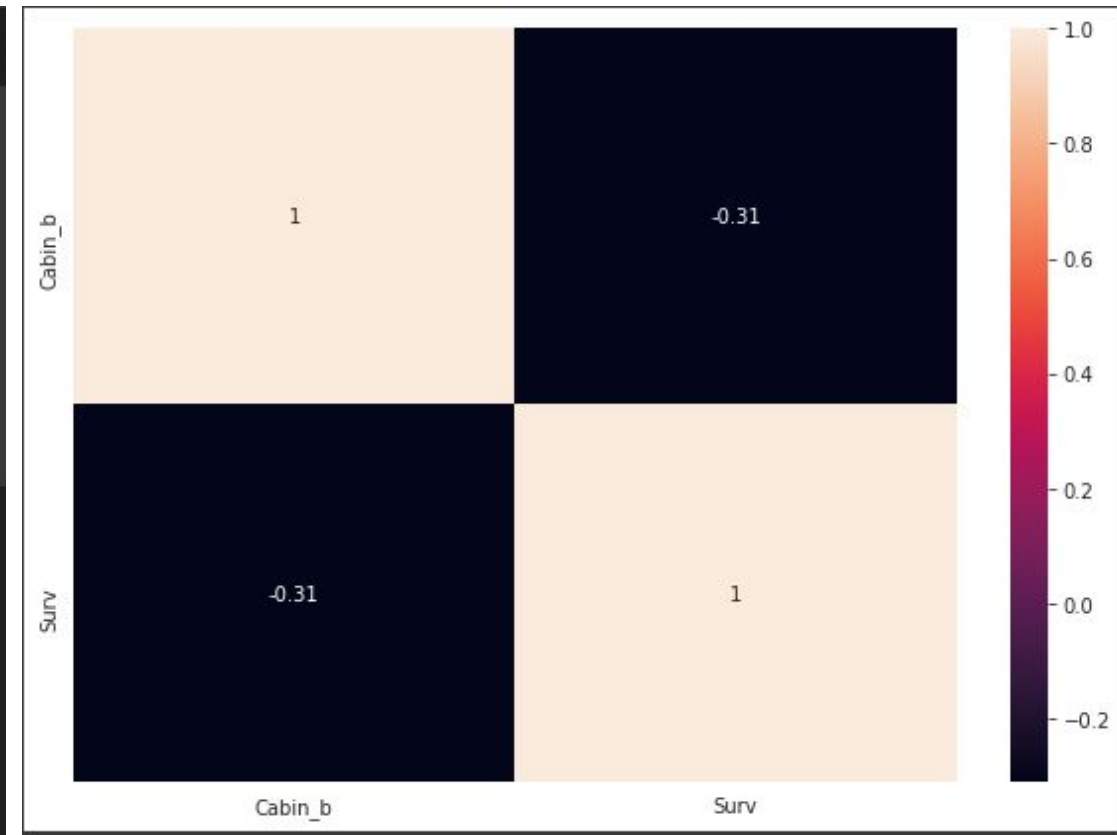
- Cabin - Номер каюты

```
[145] x_train["Cabin"].fillna('Other').apply(lambda x: str(x)[0]).value_counts()
```

```
O    544  
C     48  
B     41  
D     28  
E     25  
F     11  
A     10  
G      4  
T      1  
Name: Cabin, dtype: int64
```

```
def encode_cabin(x):  
    if str(x)[0] in ["B", "C", "D", "E", "F"]:  
        return 1  
    elif str(x)[0] in ["A", "G"]:  
        return 2  
    else:  
        return 3
```

```
x_train["Cabin_b"] = x_train["Cabin"].apply(lambda x: encode_cabin(x))  
x_test["Cabin_b"] = x_test["Cabin"].apply(lambda x: encode_cabin(x))
```



Практика. Признак Embarked.

- Embarked - Порт посадки: C = Cherbourg, Q = Queenstown, S = Southampton

```
x.Embarked.describe()
x_train["Embarked"] = x_train["Embarked"].fillna('S')

labelEncoder = LabelEncoder()

x_train['Embarked'] = labelEncoder.fit_transform(x_train['Embarked'])
x_test['Embarked'] = labelEncoder.transform(x_test['Embarked'])
x_train['Embarked'].value_counts()
```

```
2    511
0    139
1     62
Name: Embarked, dtype: int64
```


Практика. Feature Selection.

```

x_train_manual = x_train.copy()
x_test_manual = x_test.copy()
x_train_manual.drop(labels = ["Age", "SibSp", "Parch", "Fare", "Family_count", "FarePerPassenger"], axis = 1, inplace = True)
x_test_manual.drop(labels = ["Age", "SibSp", "Parch", "Fare", "Family_count", "FarePerPassenger"], axis = 1, inplace = True)
x_test_manual.head()

```

	Pclass	Sex	Embarked	Age_b	Alone	FarePerPassenger_b	Cabin_b
126	3	1	1	1	1.0	1.0	3
354	3	1	0	1	1.0	0.0	3
590	3	1	2	1	1.0	0.0	3
509	3	1	2	1	1.0	5.0	3
769	3	1	2	1	1.0	2.0	3

```

[132] logisticRegression = LogisticRegression(random_state=42)
logisticRegression.fit(x_train_manual, y_train)
y_pred = logisticRegression.predict(x_test_manual)
print("Accuracy: ", logisticRegression.score(x_test_manual, y_test))

```

Accuracy: 0.8212290502793296

Практика. Feature Selection.

```
x_train_select = x_train.copy()
x_test_select = x_test.copy()

estimator = ExtraTreesClassifier(n_estimators = 10)
featureSelection = SelectFromModel(estimator)
featureSelection.fit(x_train_select, y_train)
x_train_select = pd.DataFrame(featureSelection.transform(x_train_select), columns = x_train_select.columns.values[featureSelection.get_support()])
x_test_select = pd.DataFrame(featureSelection.transform(x_test_select), columns = x_test_select.columns.values[featureSelection.get_support()])
x_train_select.head()
```

	Sex	Age	Fare	FarePerPassenger
0	0.0	19.0	30.0000	30.0000
1	0.0	34.0	32.5000	16.2500
2	0.0	2.0	12.2875	12.2875
3	1.0	25.0	13.0000	13.0000
4	1.0	22.0	8.0500	8.0500

```
[133] logisticRegression = LogisticRegression(random_state=42)
logisticRegression.fit(x_train_select, y_train)
y_pred = logisticRegression.predict(x_test_select)
print("Accuracy: ", logisticRegression.score(x_test_select, y_test))
```

Accuracy: 0.8044692737430168

Практика. Feature Selection.



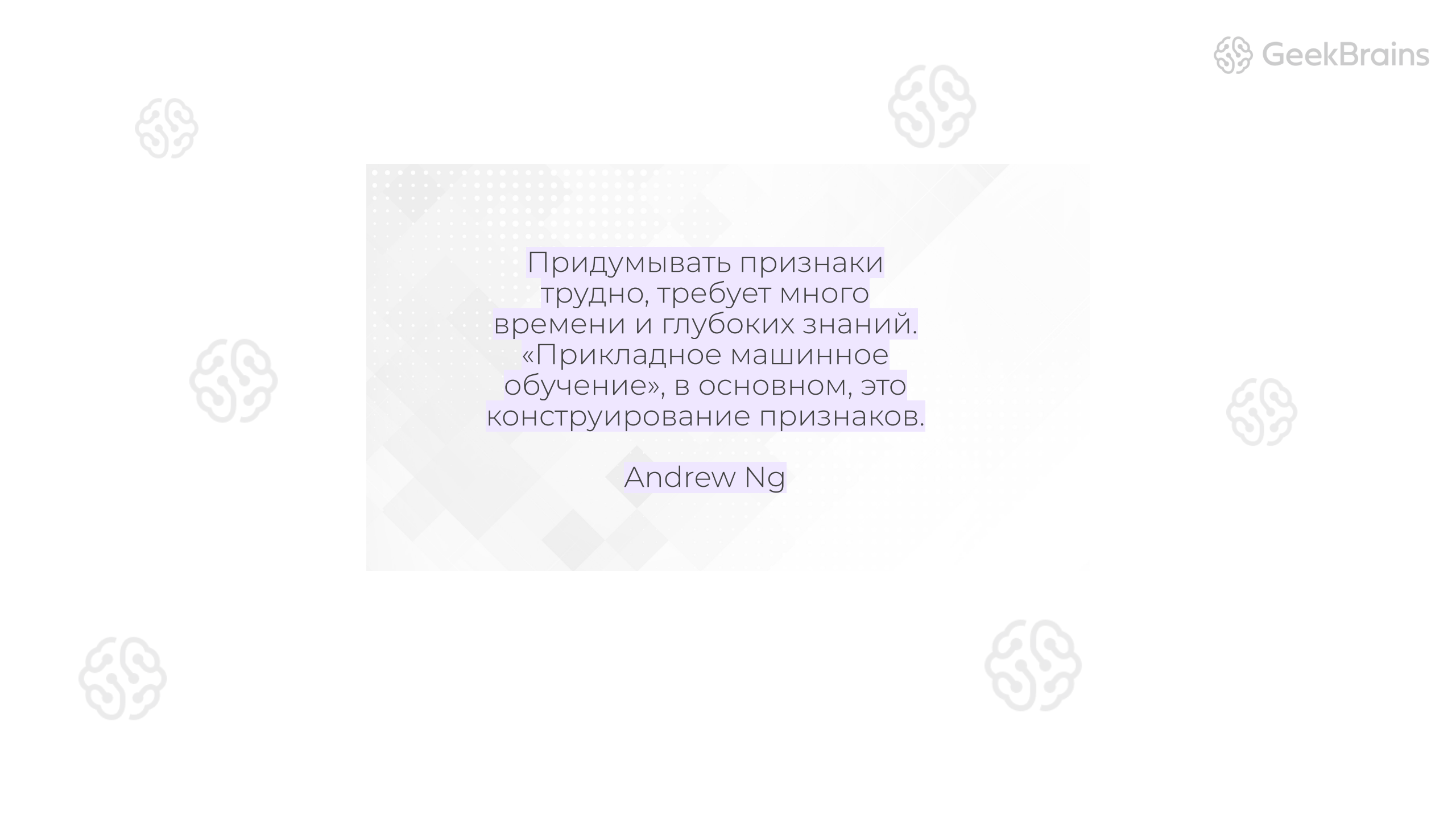
```
x_train_select2 = x_train.copy()
x_test_select2 = x_test.copy()

featureSelection = SelectKBest(chi2, k=6)
featureSelection.fit(x_train_select2, y_train)
x_train_select2 = pd.DataFrame(featureSelection.transform(x_train_select2), columns = x_train_select2.columns.values[featureSelection.get_support()])
x_test_select2 = pd.DataFrame(featureSelection.transform(x_test_select2), columns = x_test_select2.columns.values[featureSelection.get_support()])
x_train_select2.head()
```

	Pclass	Sex	Age	Fare	FarePerPassenger	FarePerPassenger_b
0	1.0	0.0	19.0	30.0000	30.0000	4.0
1	2.0	0.0	34.0	32.5000	16.2500	4.0
2	3.0	0.0	2.0	12.2875	12.2875	3.0
3	2.0	1.0	25.0	13.0000	13.0000	3.0
4	3.0	1.0	22.0	8.0500	8.0500	2.0

```
[134] logisticRegression = LogisticRegression(random_state=42)
logisticRegression.fit(x_train_select2, y_train)
y_pred = logisticRegression.predict(x_test_select2)
print("Accuracy: ", logisticRegression.score(x_test_select2, y_test))
```

Accuracy: 0.8268156424581006



Придумывать признаки
трудно, требует много
времени и глубоких знаний.
«Прикладное машинное
обучение», в основном, это
конструирование признаков.

Andrew Ng



РЕЗЮМЕ УРОКА

- Рассмотрели методы трансформации и конструирования признаков и попробовали применить их на практике
- Узнали как отбирать самые полезные признаки для улучшения точности предсказания
- Познакомились с сервисом Kaggle и поняли чем он может быть нам полезен



ИТОГОВЫЙ ПРОЕКТ

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.

Дополнительные материалы

1. <https://www.kaggle.com/c/titanic/notebooks>

ВАШИ ВОПРОСЫ