Предсказание стоимости жилья

В проекте вам нужно обучить модель линейной регрессии на данных о жилье в Калифорнии в 1990 году. На основе данных нужно предсказать медианную стоимость дома в жилом массиве. Обучите модель и сделайте предсказания на тестовой выборке. Для оценки качества модели используйте метрики RMSE, MAE и R2.

План работы

1. Инициализация локальной Spark-сессии.
2. Загрузка файла /datasets/housing.csv.
3. Выведитем типы данных колонок датасета, используя методы pySpark.
4. Предобработка данных:

- Исследование данных на наличие пропусков и заполните их.
- Преобразовние колонки с категориальными значениями техникой One hot encoding.

1. Построение двух моделей линейной регрессии на разных наборах данных:

- используя все данные из файла;
- используя только числовые переменные, исключив категориальные.

1. Анализ результатов

Загрузка и подготовка данных

In [1]:
```
import pandas as pd
import numpy as np

import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import *
import pyspark.sql.functions as F
from pyspark.ml import Pipeline

from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, StandardScaler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

In [2]:
```
RANDOM_SEED = 42

spark = SparkSession.builder \
                    .master("local") \
                    .appName("CA - Linear regression") \
                    .getOrCreate()

df = spark.read.option('header', 'true').csv('/datasets/housing.csv', inferSchema = True)
df.printSchema()
```

```
[Stage 1:>                                                          (0 + 1) / 1]
root
 |-- longitude: double (nullable = true)
 |-- latitude: double (nullable = true)
 |-- housing_median_age: double (nullable = true)
 |-- total_rooms: double (nullable = true)
 |-- total_bedrooms: double (nullable = true)
 |-- population: double (nullable = true)
 |-- households: double (nullable = true)
 |-- median_income: double (nullable = true)
 |-- median_house_value: double (nullable = true)
 |-- ocean_proximity: string (nullable = true)
```

In [3]:
```
# выведем названия колонок и тип данных
print(pd.DataFrame(df.dtypes, columns=['column', 'type']).head(10))

# выведем первые 5 строк
df.show(5)
```

```
              column    type
0          longitude  double
1           latitude  double
2  housing_median_age  double
3        total_rooms  double
4     total_bedrooms  double
5         population  double
6         households  double
7      median_income  double
8  median_house_value  double
9     ocean_proximity  string
+---------+--------+------------------+-----------+--------------+----------+----------+-------------+------------------+---------------
+
|longitude|latitude|housing_median_age|total_rooms|total_bedrooms|population|households|median_income|median_house_value|ocean_proximity
|
+---------+--------+------------------+-----------+--------------+----------+----------+-------------+------------------+---------------
+
|  -122.23|   37.88|              41.0|      880.0|         129.0|     322.0|     126.0|       8.3252|          452600.0|       NEAR BAY
|
|  -122.22|   37.86|              21.0|     7099.0|        1106.0|    2401.0|    1138.0|       8.3014|          358500.0|       NEAR BAY
|
|  -122.24|   37.85|              52.0|     1467.0|         190.0|     496.0|     177.0|       7.2574|          352100.0|       NEAR BAY
|
|  -122.25|   37.85|              52.0|     1274.0|         235.0|     558.0|     219.0|       5.6431|          341300.0|       NEAR BAY
|
|  -122.25|   37.85|              52.0|     1627.0|         280.0|     565.0|     259.0|       3.8462|          342200.0|       NEAR BAY
|
+---------+--------+------------------+-----------+--------------+----------+----------+-------------+------------------+---------------
+
only showing top 5 rows
```

In [4]:
```
# выведите базовые статистики
df.describe().toPandas()
```

Out[4]:

	summary	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	
0	count	20640	20640	20640	20640	20433	20640	20640	
1	mean	-119.56970445736148	35.6318614341087	28.639486434108527	2635.7630813953488	537.8705525375618	1425.4767441860465	499.5396802325581	3.87
2	stddev	2.003531723502584	2.135952397457101	12.58555761211163	2181.6152515827944	421.38507007403115	1132.46212176534	382.3297528316098	1.8
3	min	-124.35	32.54	1.0	2.0	1.0	3.0	1.0	
4	max	-114.31	41.95	52.0	39320.0	6445.0	35682.0	6082.0	

Итоги загрузки данных:

- 20640 наблюдений

В колонках датасета содержатся следующие данные:

- **longitude** — широта;
- **latitude** — долгота;
- **housing_median_age** — медианный возраст жителей жилого массива;
- **total_rooms** — общее количество комнат в домах жилого массива;
- **total_bedrooms** — общее количество спален в домах жилого массива;
- **population** — количество человек, которые проживают в жилом массиве;
- **households** — количество домовладений в жилом массиве;
- **median_income** — медианный доход жителей жилого массива;
- **median_house_value** — медианная стоимость дома в жилом массиве;
- **ocean_proximity** — близость к океану.

Поиск и заполнение пропусков

In [5]:
```
columns = df.columns

for column in columns:
    missing_count = df.filter(F.col(column).isNull()).count()
    if missing_count > 0:
        print(f"{column} содержит {missing_count} пропусков.")
```

```
total_bedrooms содержит 207 пропусков.
```

Заменим пропуски на средние значения

In [6]:
```
mean_bedrooms = df.agg(F.avg(df['total_bedrooms'])).first()[0]
df = df.na.fill({'total_bedrooms': mean_bedrooms})
print(f"Заменили отсутствующие значения в столбце total_bedroms на {mean_bedrooms}.")
```

```
Заменили отсутствующие значения в столбце total_bedroms на 537.8705525375618.
```

Итог раздела

- total_bedrooms содержит 207 пропусков
- Заменили отсутствующие значения в столбце total_bedroms на 537.87

Подготовка числовых и категорийных признаков

In [7]:
```
# Определение числовых и категориальных столбцов
numeric_cols = [field.name for field in df.schema.fields
                if isinstance(field.dataType, (IntegerType, FloatType, DoubleType, DecimalType))
                and field.name != 'median_house_value']
categorical_cols = [field.name for field in df.schema.fields if isinstance(field.dataType, StringType)]
target = df['median_house_value']

print("Числовые столбцы:", numeric_cols)
```

```
print("Категорийные столбцы:", categorical_cols)
print(f"Целевой столбец: median_house_value ")
```

```
Числовые столбцы: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_i
ncome']
Категорийные столбцы: ['ocean_proximity']
Целевой столбец: median_house_value
```

In [8]:
```
#Предобработка категорийных данных
indexer = StringIndexer(inputCol="ocean_proximity", outputCol="ocean_proximity_index", handleInvalid="keep")
encoder = OneHotEncoder(inputCols=["ocean_proximity_index"], outputCols=["ocean_proximity_ohe"])

# Создание ассемблеров для сборки числовых признаков в векторы
assembler_numeric_features = VectorAssembler(inputCols=numeric_cols, outputCol="features_numeric")

# Масштабирование числовых признаков
scaler = StandardScaler(inputCol="features_numeric", outputCol="scaled_features_numeric")

# Создание ассемблеров для сборки признаков в векторы
assembler_all_features = VectorAssembler(inputCols=["scaled_features_numeric"] + ['ocean_proximity_ohe'], outputCol="features_all")
```

Итог раздела

- Числовые столбцы: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']
- Категорийные столбцы: ['ocean_proximity']
- Целевой столбец: ['median_house_value']
- Проведено кодирование категориального признака через StringIndexer и OneHotEncoder
- Выполнено масштабирование признаков
- Признаки собраны в векторы

Обучение моделей

In [9]:
```
# Создание моделей линейной регрессии для обеих конфигураций признаков
lr_all_features = LinearRegression(featuresCol="features_all", labelCol="median_house_value")
lr_numeric_features = LinearRegression(featuresCol="scaled_features_numeric", labelCol="median_house_value")
```

In [10]:
```
# Пайплайн для модели со всеми признаками
pipeline_all_features = Pipeline(stages=[indexer, encoder, assembler_numeric_features,
                                         scaler, assembler_all_features, lr_all_features])

# Пайплайн для модели только с числовыми признаками
pipeline_numeric_features = Pipeline(stages=[assembler_numeric_features, scaler, lr_numeric_features])
```

In [11]:
```
# Настройка сетки параметров для кросс-валидации
paramGrid = ParamGridBuilder() \
    .addGrid(lr_all_features.regParam, [0.1, 0.01]) \
    .addGrid(lr_all_features.elasticNetParam, [0.0, 0.5, 1.0]) \
    .build()

crossval_all_features = CrossValidator(estimator=pipeline_all_features,
                                       estimatorParamMaps=paramGrid,
                                       evaluator=RegressionEvaluator(labelCol="median_house_value"),
                                       numFolds=5)

crossval_numeric_features = CrossValidator(estimator=pipeline_numeric_features,
                                           estimatorParamMaps=paramGrid,
                                           evaluator=RegressionEvaluator(labelCol="median_house_value"),
                                           numFolds=5)
```

In [12]:
```
# Разделение данных на обучающий и тестовый наборы
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)
```

In [13]:
```
# Обучение и оценка модели со всеми признаками
cvModel_all_features = crossval_all_features.fit(train_data)
predictions_all_features = cvModel_all_features.transform(test_data)
```

```
24/05/03 09:32:48 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
24/05/03 09:32:48 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
24/05/03 09:32:49 WARN LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
24/05/03 09:32:49 WARN LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
```

In [14]:
```
# Обучение и оценка модели только с числовыми признаками
cvModel_numeric_features = crossval_numeric_features.fit(train_data)
predictions_numeric_features = cvModel_numeric_features.transform(test_data)
```

```
24/05/03 09:33:47 WARN Instrumentation: [b62774de] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:48 WARN Instrumentation: [0474eecf] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:49 WARN Instrumentation: [3fa7d604] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:50 WARN Instrumentation: [3a4496cf] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:50 WARN Instrumentation: [8d327256] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:51 WARN Instrumentation: [2dec20dc] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:52 WARN Instrumentation: [30adaa91] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:53 WARN Instrumentation: [61ca6af3] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:54 WARN Instrumentation: [756f2ad1] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:55 WARN Instrumentation: [76f82cd5] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:55 WARN Instrumentation: [0783a3e0] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:56 WARN Instrumentation: [03e656ad] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:57 WARN Instrumentation: [7e8d47b3] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:58 WARN Instrumentation: [7ad083c5] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:59 WARN Instrumentation: [ffdb764c] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:33:59 WARN Instrumentation: [806bfc52] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:00 WARN Instrumentation: [33e2fa2b] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:01 WARN Instrumentation: [b5bacc83] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:02 WARN Instrumentation: [76730c78] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:02 WARN Instrumentation: [cfc62246] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:03 WARN Instrumentation: [66a9eff7] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:04 WARN Instrumentation: [e14e8814] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:04 WARN Instrumentation: [4c9b5136] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:05 WARN Instrumentation: [ed60125a] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:06 WARN Instrumentation: [4b640c5a] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:07 WARN Instrumentation: [73b54533] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:08 WARN Instrumentation: [193a614f] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:08 WARN Instrumentation: [f2e9b094] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:09 WARN Instrumentation: [e145cbf9] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:10 WARN Instrumentation: [faf2e219] regParam is zero, which might cause numerical instability and overfitting.
24/05/03 09:34:11 WARN Instrumentation: [106db315] regParam is zero, which might cause numerical instability and overfitting.
```

In [15]:
```
# Создание оценщиков для метрик качества модели
evaluator_rmse = RegressionEvaluator(labelCol="median_house_value", metricName="rmse")
evaluator_mae = RegressionEvaluator(labelCol="median_house_value", metricName="mae")
evaluator_r2 = RegressionEvaluator(labelCol="median_house_value", metricName="r2")
```

In [16]:
```
# Расчёт метрик для каждой модели
rmse_all = evaluator_rmse.evaluate(predictions_all_features)
mae_all = evaluator_mae.evaluate(predictions_all_features)
r2_all = evaluator_r2.evaluate(predictions_all_features)

rmse_numeric = evaluator_rmse.evaluate(predictions_numeric_features)
mae_numeric = evaluator_mae.evaluate(predictions_numeric_features)
r2_numeric = evaluator_r2.evaluate(predictions_numeric_features)
```

In [17]:
```
print("Результаты модели для всех признаков:")
print("RMSE:", rmse_all)
print("MAE:", mae_all)
print("R^2:", r2_all)

print("\nРезультаты модели для числовых признаков:")
print("RMSE:", rmse_numeric)
print("MAE:", mae_numeric)
print("R^2:", r2_numeric)
```

```
Результаты модели для всех признаков:
RMSE: 70781.4840513847
MAE: 50854.202700630536
R^2: 0.6378961901340965

Результаты модели для числовых признаков:
RMSE: 71782.88243179358
MAE: 51787.84444119117
R^2: 0.6275778072485874
```

In [18]:
```
#Отключаем spark сессию
spark.stop()
```

Анализ результатов

Итоги по ходу проекта:

Итоги загрузки данных:

- 20640 наблюдений

В колонках датасета содержатся следующие данные:

- **longitude** — широта;
- **latitude** — долгота;
- **housing_median_age** — медианный возраст жителей жилого массива;
- **total_rooms** — общее количество комнат в домах жилого массива;
- **total_bedrooms** — общее количество спален в домах жилого массива;
- **population** — количество человек, которые проживают в жилом массиве;
- **households** — количество домовладений в жилом массиве;
- **median_income** — медианный доход жителей жилого массива;
- **median_house_value** — медианная стоимость дома в жилом массиве;
- **ocean_proximity** — близость к океану.

Итог проверки данных на пропуски:

- total_bedrooms содержит 207 пропусков
- Заменили отсутствующие значения в столбце total_bedroms на 537.87

Итог подготовки данных:

- Числовые столбцы: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']

- Категорийные столбцы: ['ocean_proximity']
- Целевой столбец: ['median_house_value']
- Проведено кодирование категориального признака через StringIndexer и OneHotEncoder
- Выполнено масштабирование признаков
- Признаки собраны в векторы

Итоги обучения модели:

- Построен пайплайн для обработки данных под два вида моделей **Настроена сетка параметров:**
- regParam: Этот параметр в модели линейной регрессии относится к параметру регуляризации. Регуляризация помогает предотвратить переобучение модели за счёт добавления штрафа за слишком большие веса в модели. Значения [0.1, 0.01] указывают, что в процессе кросс-валидации будут протестированы два уровня силы регуляризации: 0.1 и 0.01, где меньшее значение означает более слабую регуляризацию.
- elasticNetParam: Этот параметр определяет сочетание L1 и L2 регуляризации в модели, что часто называется Elastic Net регуляризацией. Значение 0.0 соответствует чистой L2 регуляризации (также известной как Ridge), 1.0 соответствует чистой L1 регуляризации (также известной как Lasso), а 0.5 представляет собой равное сочетание L1 и L2 регуляризации.

Анализ результатов:

Модель со всеми признаками (включая категориальные, преобразованные через One Hot Encoding):

- RMSE (Root Mean Squared Error, Среднеквадратичная ошибка): 70781.23
- MAE (Mean Absolute Error, Средняя абсолютная ошибка): 50855.06
- R^2 (коэффициент детерминации): 0.638 Эта модель показывает лучшее качество прогнозов по всем трем метрикам. Ниже значение RMSE и MAE свидетельствует о том, что модель в среднем делает меньшие ошибки в предсказаниях. Более высокое значение R^2 указывает на то, что модель лучше объясняет вариативность наблюдаемых данных.

Модель только с числовыми признаками:

- RMSE: 71782.88
- MAE: 51787.84
- R^2: 0.628 Модель с числовыми признаками показывает худшие результаты по сравнению с моделью, включающей все признаки. Это может означать, что категориальные признаки вносят значимый вклад в точность и объясняемость модели. Выводы

Влияние категориальных признаков: Присутствие категориальных признаков, преобразованных через метод One Hot Encoding, значительно улучшает производительность модели. Это подтверждается улучшением всех трех метрик (RMSE, MAE и R^2), что подчеркивает их важность для создания точной предиктивной модели в данной задаче.

Выбор модели для разработки: На основе этих результатов рекомендуется использовать модель со всеми признаками для дальнейшей разработки и оптимизации, поскольку она показывает лучшую способность к генерализации и предсказанию.

Возможности улучшения: Несмотря на то, что модель со всеми признаками показывает лучшие результаты, есть потенциал для дальнейшего улучшения, например, путем тонкой настройки гиперпараметров, использования более сложных методов моделирования.