```
]: import pandas as pd

import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler
import numpy as np

# Загрузка датасета
car_data = pd.read_csv('Car_sales.csv')

# Вывод первых нескольких строк датасета
print(car_data.head())

# Проверка наличия пропущенных значений
print("\nПропущенные значения в датасете:")
print(car_data.isnull().sum())

car_data.describe()
print(len(car_data))
```

```
  Manufacturer    Model  Sales_in_thousands  __year_resale_value Vehicle_type  \
0        Acura  Integra              16.919                16.360    Passenger
1        Acura       TL              39.384                19.875    Passenger
2        Acura       CL              14.114                18.225    Passenger
3        Acura       RL               8.588                29.725    Passenger
4         Audi       A4              20.397                22.255    Passenger

   Price_in_thousands  Engine_size  Horsepower  Wheelbase  Width  Length  \
0               21.50          1.8       140.0      101.2   67.3   172.4
1               28.40          3.2       225.0      108.1   70.3   192.9
2                 NaN          3.2       225.0      106.9   70.6   192.0
3               42.00          3.5       210.0      114.6   71.4   196.6
4               23.99          1.8       150.0      102.6   68.2   178.0

   Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
0        2.639           13.2             28.0      2/2/2012
1        3.517           17.2             25.0      6/3/2011
2        3.470           17.2             26.0      1/4/2012
3        3.850           18.0             22.0     3/10/2011
4        2.998           16.4             27.0     10/8/2011

   Power_perf_factor
0          58.280150
1          91.370778
2                NaN
3          91.389779
4          62.777639

Пропущенные значения в датасете:
Manufacturer           0
Model                  0
Sales_in_thousands     0
__year_resale_value    36
Vehicle_type           0
Price_in_thousands     2
Engine_size            1
Horsepower             1
Wheelbase              1
Width                  1
Length                 1
Curb_weight            2
Fuel_capacity          1
Fuel_efficiency        3
Latest_Launch          0
Power_perf_factor      2
dtype: int64
157
```

```
# Обработка пропусков в данных
# Заполнение пропущенных значений медианными значениями
car_data['__year_resale_value'].fillna(car_data['__year_resale_value'].median(), inplace=True)
car_data['Price_in_thousands'].fillna(car_data['Price_in_thousands'].median(), inplace=True)
car_data['Engine_size'].fillna(car_data['Engine_size'].median(), inplace=True)
car_data['Horsepower'].fillna(car_data['Horsepower'].median(), inplace=True)
car_data['Wheelbase'].fillna(car_data['Wheelbase'].median(), inplace=True)
car_data['Width'].fillna(car_data['Width'].median(), inplace=True)
car_data['Length'].fillna(car_data['Length'].median(), inplace=True)
car_data['Curb_weight'].fillna(car_data['Curb_weight'].median(), inplace=True)
car_data['Fuel_capacity'].fillna(car_data['Fuel_capacity'].median(), inplace=True)
car_data['Fuel_efficiency'].fillna(car_data['Fuel_efficiency'].median(), inplace=True)
car_data['Power_perf_factor'].fillna(car_data['Power_perf_factor'].median(), inplace=True)
```

```
print("\nПроверка значений пропущенных в датасете:")
print(car_data.isnull().sum())
```

```
Проверка значений пропущенных в датасете:
Manufacturer          0
Model                 0
Sales_in_thousands    0
__year_resale_value   0
Vehicle_type          0
Price_in_thousands    0
Engine_size           0
Horsepower            0
Wheelbase             0
Width                 0
Length                0
Curb_weight           0
Fuel_capacity         0
Fuel_efficiency       0
Latest_Launch         0
Power_perf_factor     0
dtype: int64
```

```
# Кодирование категориальных признаков
# One-Hot Encoding для признака "Vehicle_type"
vehicle_type_encoder = OneHotEncoder(sparse=False)
vehicle_type_encoded = vehicle_type_encoder.fit_transform(car_data[['Vehicle_type']])
vehicle_type_encoded_df = pd.DataFrame(vehicle_type_encoded, columns=vehicle_type_encoder.categories_[0])

# Добавление закодированных признаков к исходному датасету
car_data = pd.concat([car_data, vehicle_type_encoded_df], axis=1)

# Удаление исходного признака "Vehicle_type"
car_data.drop('Vehicle_type', axis=1, inplace=True)
```

```
# Масштабирование числовых признаков
scaler = StandardScaler()
car_data[['Horsepower']] = scaler.fit_transform(car_data[['Horsepower']])


# Вывод первых нескольких строк обработанного датасета
print(car_data.head())
```

```
  Manufacturer    Model  Sales_in_thousands  __year_resale_value  \
0        Acura  Integra              16.919                16.360
1        Acura       TL              39.384                19.875
2        Acura       CL              14.114                18.225
3        Acura       RL               8.588                29.725
4         Audi       A4              20.397                22.255

   Price_in_thousands  Engine_size  Horsepower  Wheelbase  Width  Length  \
0              21.500          1.8   -0.814577      101.2   67.3   172.4
1              28.400          3.2    0.694066      108.1   70.3   192.9
2              22.799          3.2    0.694066      106.9   70.6   192.0
3              42.000          3.5    0.427835      114.6   71.4   196.6
4              23.990          1.8   -0.637089      102.6   68.2   178.0

   Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
0        2.639           13.2             28.0      2/2/2012
1        3.517           17.2             25.0      6/3/2011
2        3.470           17.2             26.0      1/4/2012
3        3.850           18.0             22.0     3/10/2011
4        2.998           16.4             27.0     10/8/2011

   Power_perf_factor  Car  Passenger
0          58.280150  0.0        1.0
1          91.370778  0.0        1.0
2          72.030917  0.0        1.0
3          91.389779  0.0        1.0
4          62.777639  0.0        1.0
```

```
from sklearn.model_selection import train_test_split

# Преобразование Sales_in_thousands в категориальный признак
car_data['Sales_category'] = pd.cut(car_data['Sales_in_thousands'],
                                    bins=[-np.inf, car_data['Sales_in_thousands'].mean(), np.inf],
                                    labels=['Low', 'High'])

# Удалим признаки, которые не будем использовать для обучения модели
car_data.drop(['Manufacturer', 'Model', 'Sales_in_thousands'], axis=1, inplace=True)

# Разделим данные на признаки и целевую переменную
X = car_data.drop('Sales_category', axis=1)
y = car_data['Sales_category']

# Кодирование категориальных признаков
X_encoded = pd.get_dummies(X, drop_first=True)

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)
```

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Инициализация и обучение модели ближайших соседей
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Предсказание на тестовых данных
y_pred = knn.predict(X_test)

# Оценка качества модели
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy на тестовой выборке для K=5:", accuracy)
```

```
/lib/python3.11/site-packages/threadpoolctl.py:1019: Runti
s OS.
  warnings.warn(
Accuracy на тестовой выборке для K=5: 0.75
```

```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

# Определение диапазона значений K
param_grid = {'n_neighbors': range(1, 21)}

# Инициализация GridSearchCV
grid_search = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Лучшее значение гиперпараметра K
best_k = grid_search.best_params_['n_neighbors']
print("Лучшее значение K:", best_k)

# Оценка качества модели с лучшим значением K
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_test)
accuracy_best = accuracy_score(y_test, y_pred_best)
print("Accuracy на тестовой выборке с оптимальным K:", accuracy_best)
```

```
Лучшее значение K: 19
Accuracy на тестовой выборке с оптимальным K: 0.8125
```

```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

# Определение диапазона значений K
param_grid = {'n_neighbors': range(1, 21)}

# Инициализация GridSearchCV
grid_search = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Лучшее значение гиперпараметра K
best_k = grid_search.best_params_['n_neighbors']
print("Лучшее значение K:", best_k)
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_train)
accuracy_best = accuracy_score(y_train, y_pred_best)
print("Accuracy на тренировочной выборке с оптимальным K:", accuracy_best)
```

```
Лучшее значение K: 19
Accuracy на тренировочной выборке с оптимальным K: 0.68
```

```python
# Инициализация RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=KNeighborsClassifier(), param_distributions=param_grid, n_iter=10, cv=5, scoring='accuracy', rand
random_search.fit(X_train, y_train)

# Лучшее значение гиперпараметра K
best_k_random = random_search.best_params_['n_neighbors']
print("Лучшее значение K (RandomizedSearchCV):", best_k_random)

# Оценка качества модели с лучшим значением K
best_knn_random = random_search.best_estimator_
y_pred_best_random = best_knn_random.predict(X_test)
accuracy_best_random = accuracy_score(y_test, y_pred_best_random)
print("Accuracy на тестовой выборке с оптимальным K (RandomizedSearchCV):", accuracy_best_random)
```

```
Лучшее значение K (RandomizedSearchCV): 19
Accuracy на тестовой выборке с оптимальным K (RandomizedSearchCV): 0.8125
```