«Машинное обучение в задачах обработки финансовой и экономической информации»

"Машинное обучение в задачах обработки финансовой и экономической информации"

Цель работы: исследование и обработки финансовой и экономической информации для прогнозирования и анализа выручки предприятия с помощью различных методов машинного обучения

Задачи:

1) Сбор данных о выручке предприятия

2) Исследование данных

3) Построение модели машинного обучения

4) Оценка и анализ результатов прогнозирования

Результат работы: спрогнозированное значение выручки предприятия

Описание датасета: в датасете представлены данные о выручки предприятия, со следующими параметрами:

- Store - идентификатор магазина
- Dept - идентификатор отдела
- Date - дата
- Weekly_Sales- выручка за неделю
- IsHoliday- является день праздничным или нет
- Item_Identifier – идентификатор продукта
- Item_Weight – вес продукта
- Item_Fat_Content – содержание жиров в продукте ("Low Fat" или "Regular")
- Item_Visibility – доля площади, занимаемой продуктом на полке среди всех продуктов
- Item_Type – тип продукта
- Item_MRP – максимальная розничная цена продукта
- Outlet_Identifier – идентификатор магазина
- Outlet_Establishment_Year – год открытия магазина
- Outlet_Size – размер магазина
- Outlet_Location_Type – тип местоположения магазина ("Tier 1", "Tier 2" или "Tier 3")
- Outlet_Type – тип магазина ("Supermarket Type1", "Supermarket Type2", "Supermarket Type3" или "Grocery Store")
- Item_Outlet_Sales – продажи продукта в конкретном магазине

Введение

Тема "Методы машинного обучения для прогнозирования и анализа выручки предприятия" является весьма актуальной в настоящее время, поскольку выручка является одним из ключевых показателей финансовой эффективности предприятия. Предприятия, которые могут точно прогнозировать свою будущую выручку, могут лучше планировать свою деятельность и принимать более обоснованные решения по расходам и инвестициям.

С использованием методов машинного обучения можно создать модели, которые могут прогнозировать будущую выручку на основе исторических данных о продажах и других факторах. Эти модели могут помочь предприятиям оптимизировать свои бизнес-процессы, улучшить управление запасами и планирование производства, а также принимать обоснованные решения по увеличению выручки и прибыли.

В целом, использование методов машинного обучения для прогнозирования и анализа выручки предприятия может помочь компаниям улучшить свою финансовую эффективность, увеличить прибыль и обеспечить более эффективное управление бизнесом.

Основная часть

In [44]:
```
# импортируем необходимые библиотеки
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import PassiveAggressiveRegressor
from sklearn.model_selection import train_test_split
from sklearn import linear_model, model_selection
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import KFold,StratifiedKFold,LeaveOneOut, cross_val_score
from datetime import datetime
import time
import warnings
warnings.filterwarnings("ignore")
```

Загружаю данные

In [2]:
```
df = pd.read_csv('my_dataframe.csv')
df.head()
```

Out[2]:

	Dept	Date	Weekly_Sales	IsHoliday	Stores	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
0	1	2010-02-05	24924.50	False	1	FDA15	9.30	Low Fat	0.016047
1	1	2010-02-12	46039.49	True	2	DRC01	5.92	Regular	0.019278

	Dept	Date	Weekly_Sales	IsHoliday	Stores	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
2	1	2010-19	41595.55	False	3	FDN15	17.50	Low Fat	0.016760
3	1	2010-26	19403.54	False	4	FDX07	19.20	Regular	0.000000
4	1	2010-03-05	21827.90	False	5	NCD19	8.93	Low Fat	0.000000

Предварительный анализ и очистка данных

In [3]:
```
print("Количество строк и столбцов в датасете:")
df.shape
```

```
Количество строк и столбцов в датасете:
```

Out[3]:
```
(8523, 17)
```

In [4]:
```
print("Типы данных в датасете:")
df.dtypes
```

```
Типы данных в датасете:
```

Out[4]:
```
Dept                         int64
Date                        object
Weekly_Sales               float64
IsHoliday                     bool
Stores                       int64
Item_Identifier             object
Item_Weight                float64
Item_Fat_Content            object
Item_Visibility            float64
Item_Type                   object
Item_MRP                   float64
Outlet_Identifier           object
Outlet_Establishment_Year    int64
Outlet_Size                 object
Outlet_Location_Type        object
Outlet_Type                 object
Item_Outlet_Sales          float64
dtype: object
```

In [5]:
```
print('Проверим на пустые значения:')
df.isna().sum()
```

```
Проверим на пустые значения:
```

Out[5]:
```
Dept                          0
Date                          0
Weekly_Sales                  0
IsHoliday                     0
Stores                        0
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

In [6]:
```
df.fillna(0, inplace=True)
```

In [7]:
```
print('Проверим на пустые значения:')
df.isna().sum()
```

```
Проверим на пустые значения:
```

Out[7]:
```
Dept                          0
Date                          0
Weekly_Sales                  0
IsHoliday                     0
Stores                        0
Item_Identifier               0
Item_Weight                   0
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                   0
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

Видим, что пустые значения отсутсвуют

In [8]:
```
# Вывод статистических характеристик датасета
df.describe()
```

Out[8]:

	Dept	Weekly_Sales	Stores	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Y
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000
mean	34.796668	14884.621963	0.022293	10.650590	0.066132	140.992782	1997.831
std	22.800247	17135.493279	0.537904	6.431899	0.051598	62.275067	8.371
min	1.000000	-863.000000	0.000000	0.000000	0.000000	31.290000	1985.000
25%	16.000000	2855.555000	0.000000	6.650000	0.026989	93.826500	1987.000
50%	32.000000	8445.010000	0.000000	11.000000	0.053931	143.012800	1999.000
75%	52.000000	20625.965000	0.000000	16.000000	0.094585	185.643700	2004.000
max	83.000000	203670.470000	19.000000	21.350000	0.328391	266.888400	2009.000

In [9]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

Преобразование атрибутов исходного датасетаа

Заменяю категориальные значения на числа

1. Outlet_Size:

   "Small" - 1\ "Medium" - 2\ "High" - 3

1. Outlet_Location_Type:

   "Tier 1" - 1\ "Tier 2" - 2\ "Tier 3" - 3

1. Outlet_Type:

   "Supermarket Type1" - 1\ "Supermarket Type2" - 2\ "Supermarket Type3" - 3\ "Grocery Store" -> 4

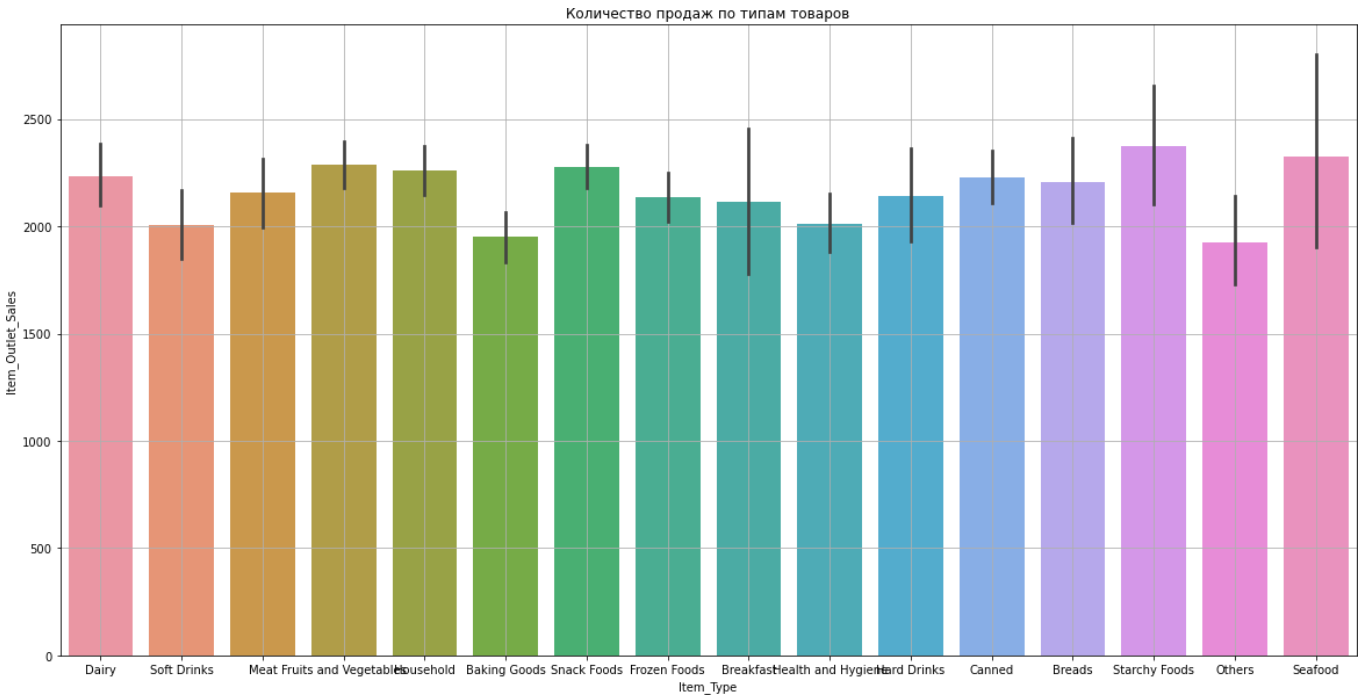
1. Item_Fat_Content:

   "Regular" - 1\ "Low Fat" - 2

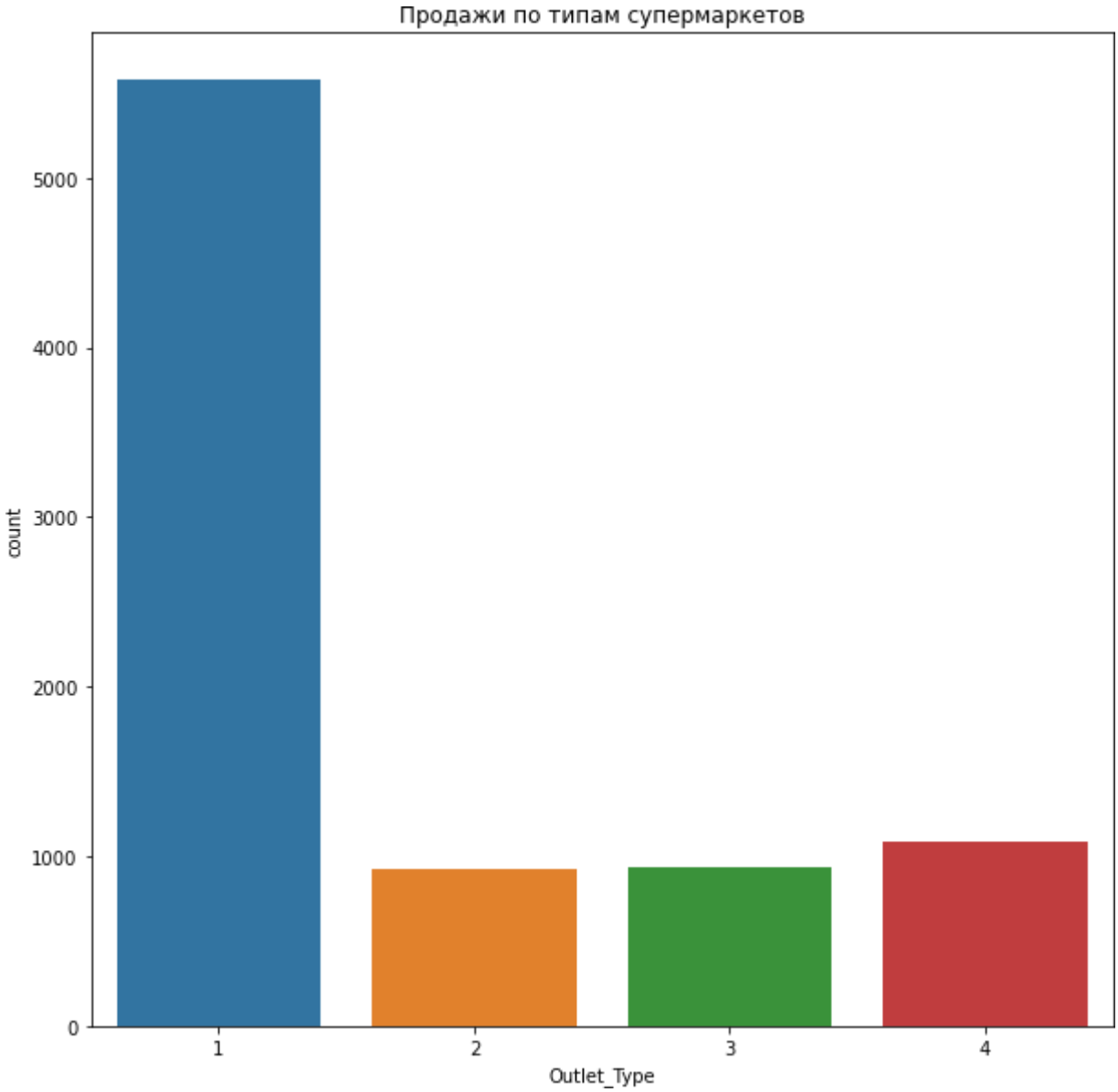
In [10]:
```
df["Outlet_Size"] = df["Outlet_Size"].replace({"Small": 1, "Medium": 2, "High": 3})
df["Item_Fat_Content"] = df["Item_Fat_Content"].replace({"Regular" : 1, "Low Fat" : 2,"low fat
df["Outlet_Location_Type"] = df["Outlet_Location_Type"].replace({"Tier 1": 1, "Tier 2": 2, "Ti
df["Outlet_Type"] =  df["Outlet_Type"].replace({"Supermarket Type1": 1, "Supermarket Type2": 2,
```

Визуализация данных

In [11]:
```
plt.figure(figsize=(20,10))
sns.barplot(x='Item_Type',y='Item_Outlet_Sales',data=df)
plt.title('Количество продаж по типам товаров')
plt.grid()
```

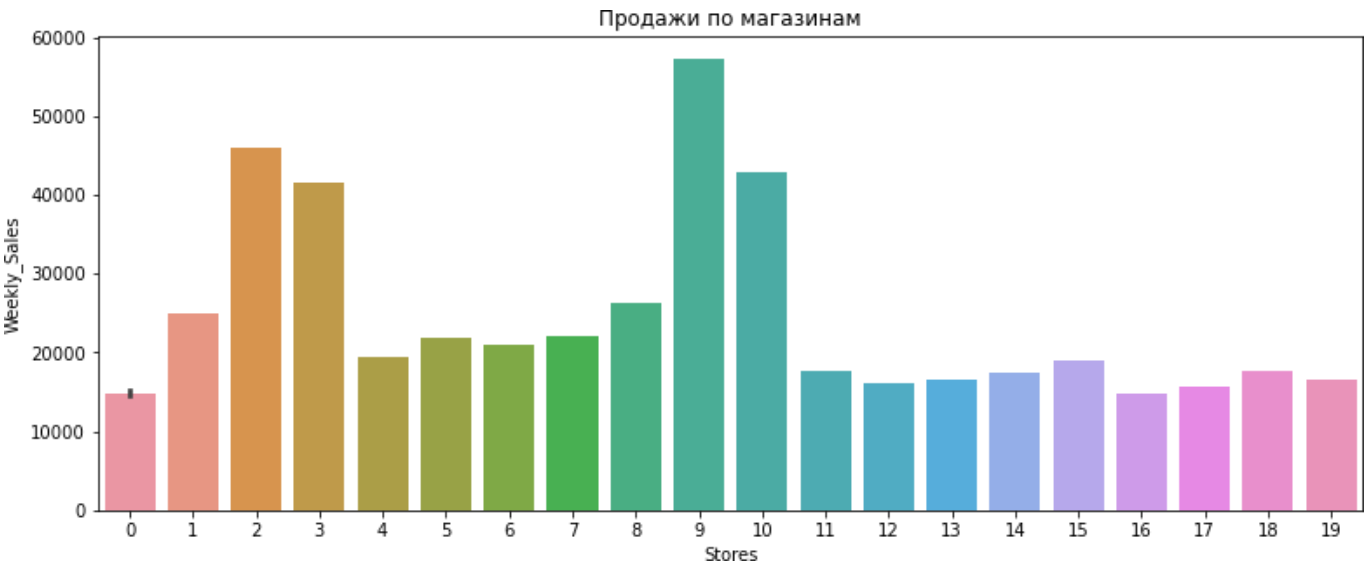


In [12]:
```
plt.figure(figsize=(10,10))
sns.countplot(x="Outlet_Type",  data=df)
plt.title('Продажи по типам супермаркетов')
plt.show()
```



In [13]:

```
plt.figure(figsize = (13,5))
sns.barplot(x='Stores', y='Weekly_Sales', data=df)
plt.title('Продажи по магазинам')
plt.show()
```



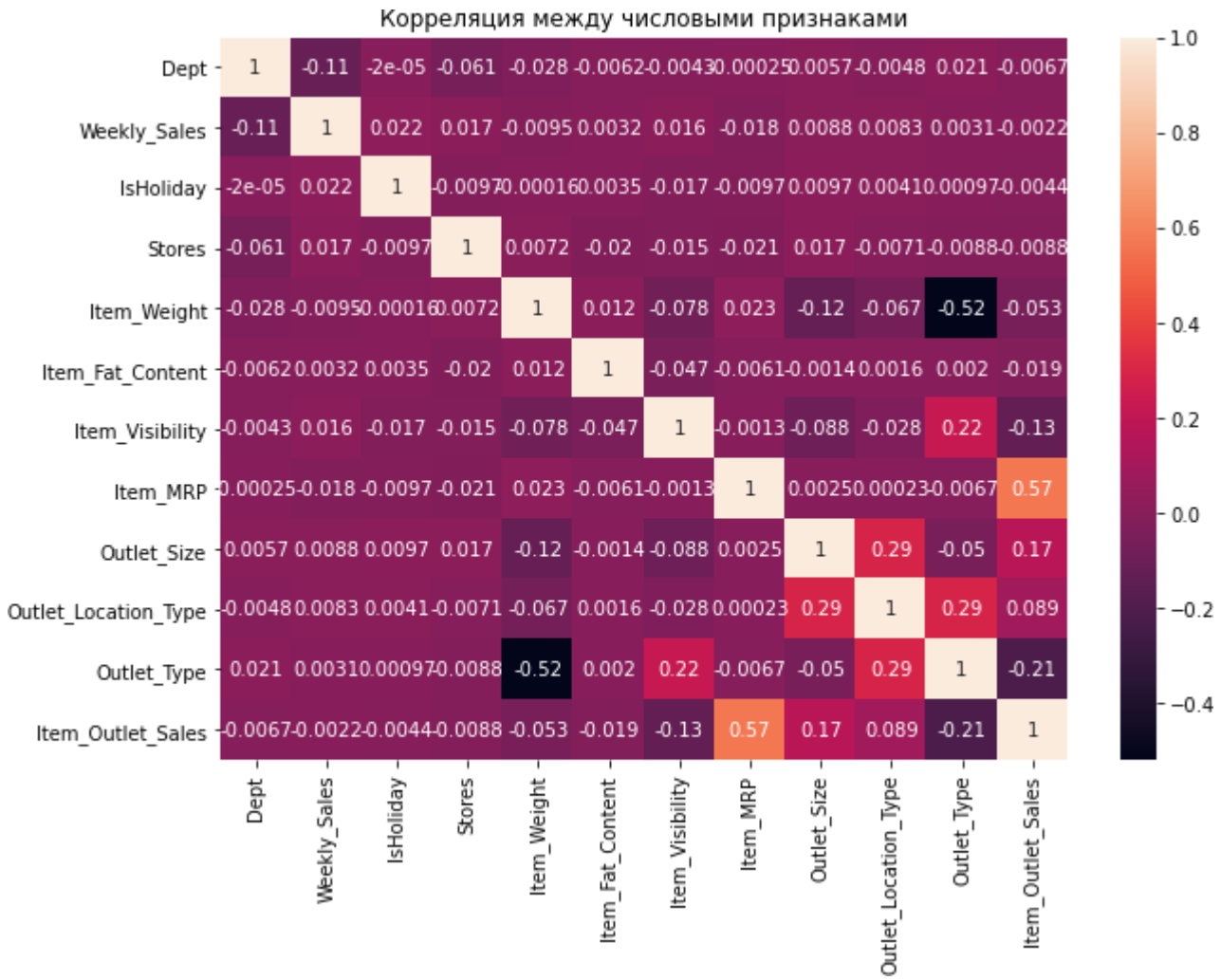
In [14]:

```
plt.figure(figsize = (7,5))
sns.lineplot(x='Date', y='Weekly_Sales', data=df)
plt.title('Выручка по датам')
plt.show()
```

Удаляем ненужные стобцы

In [15]:
```
df=df.drop(['Item_Identifier','Outlet_Establishment_Year','Outlet_Identifier','Item_Type','Dat
```

In [16]:
```
plt.figure(figsize = (10,7))
numeric_data = df
corr = numeric_data.corr()
sns.heatmap(corr, annot=True)
plt.title('Корреляция между числовыми признаками')
plt.show()
```



In [17]:
```
df.head()
```

Out[17]:

	Dept	Weekly_Sales	IsHoliday	Stores	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Size
0	1	24924.50	False	1	9.30	2	0.016047	249.8092	2
1	1	46039.49	True	2	5.92	1	0.019278	48.2692	2
2	1	41595.55	False	3	17.50	2	0.016760	141.6180	2
3	1	19403.54	False	4	19.20	1	0.000000	182.0950	0
4	1	21827.90	False	5	8.93	2	0.000000	53.8614	3

Модели машиного обучения для прогнозирования выручки :

Разделяю набор данных на обучающую и тестовую выборки

In [18]:
```
X = df.drop(['Weekly_Sales','Item_Outlet_Sales'], axis=1)
y = df['Weekly_Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Количественные характеристики разбиения данных на выборки:

- Обучающая выборка: 80% данных
- Тестовая выборка: 20% данных

Метод разделения данных: случайный.

Линейная регрессия

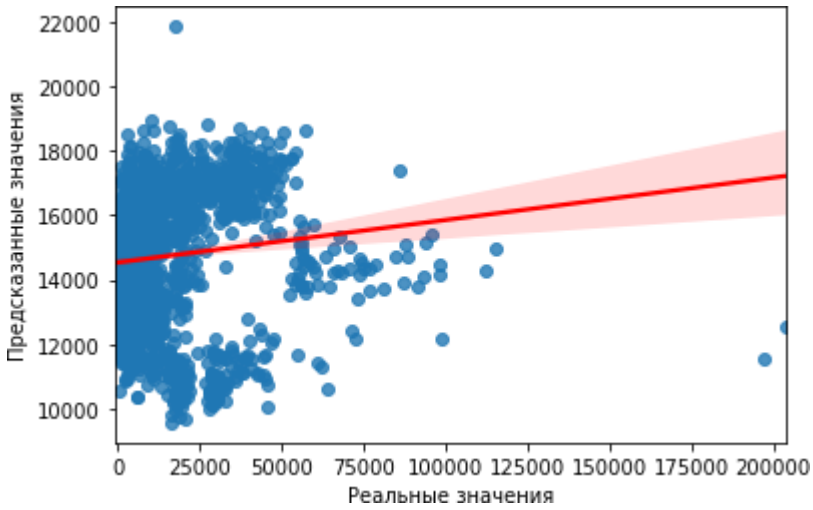
In [47]:
```
%%time
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('RMSE:', rmse)
print('MAE:', mae)
print('R2:', r2)
```

```
RMSE: 17922.97373083954
MAE: 12699.071337385229
R2: 0.011511909142914445
Wall time: 19.5 ms
```

Значение RMSE достаточно высокое (17929.45), что означает, что среднеквадратичная ошибка предсказания модели достаточно большая. Значение MAE (также довольно высокое, что указывает на большую погрешность предсказания. Значение коэффициента детерминации R2 очень близко к нулю (0.0115), что говорит о том, что модель не применима для нашей работы.

Точность модели крайне низкая ,график выглядит следующим образом образом .

In [48]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Дерево решений

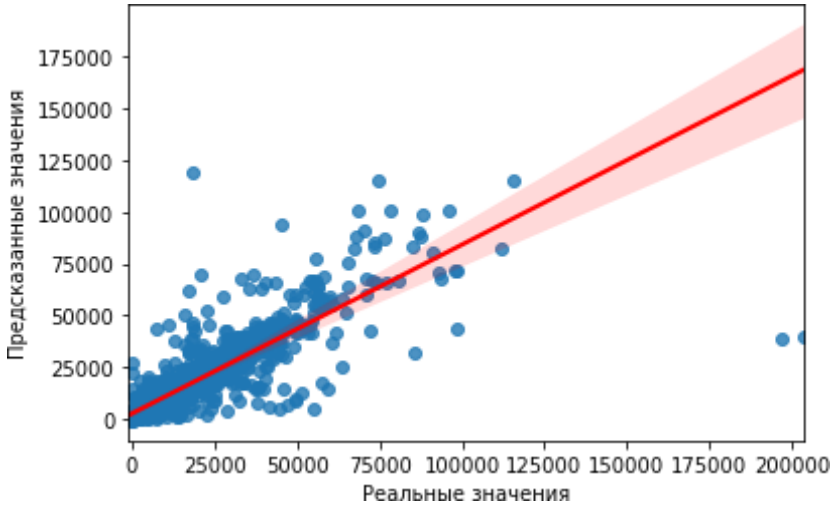
In [49]:
```
%%time
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
y_pred = dtr.predict(X_test)
rmse_dtr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_dtr= mean_absolute_error(y_test, y_pred)
r2_dtr = r2_score(y_test, y_pred)
print('RMSE:', rmse_dtr)
print('MAE:', mae_dtr)
print('R2:', r2_dtr)
```

```
RMSE: 9442.221203083489
MAE: 3854.0669442815247
R2: 0.725653359940403
Wall time: 90.7 ms
```

Значение RMSE стало значительно меньше, что говорит о том, что качество предсказаний модели значительно улучшилось после подбора оптимальных параметров. Значение MAE также уменьшилось, что указывает на уменьшение погрешности предсказания. Значение коэффициента детерминации R2 значительно увеличилось и достигло величины 0.736, что указывает на то, что модель значительно лучше предсказывает значения.

Точность модели достаточно высокая , модель предсказывает хорошо. Визуализируем полученные результаты :

In [50]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Случайный лес

In [51]:

```
%%time
rfr=RandomForestRegressor()
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
```

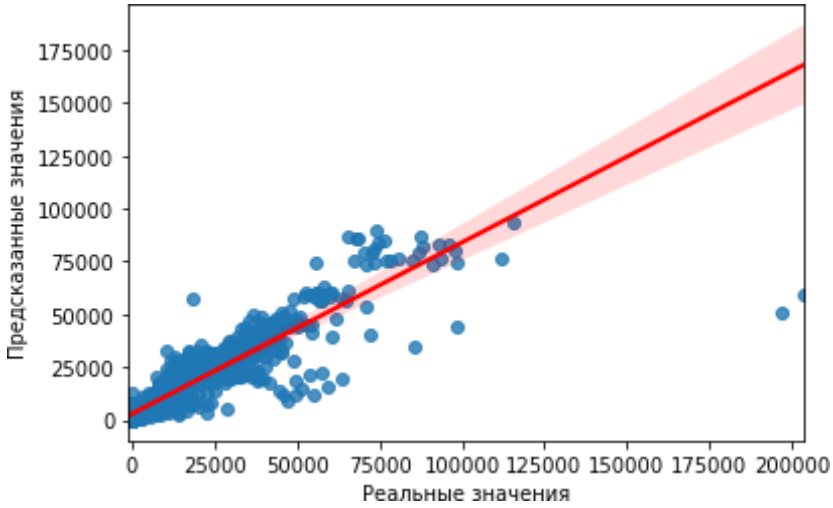
```
RMSE: 7581.971639566451
MAE: 3035.218150381232
R2: 0.8231049761794419
Wall time: 7.79 s
```

Значение RMSE стало еще меньше, чем в предыдущих случаях, что указывает на еще более точное предсказание модели. Значение MAE также уменьшилось, что указывает на еще меньшую погрешность предсказания. Значение коэффициента детерминации R2 очень близко к единице (0.823), что указывает на то, что модель отлично предсказывает значения.

Точность модели достаточно высокая , модель предсказывает хорошо. Визуализируем полученные результаты :

In [52]:

```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Бэггинг

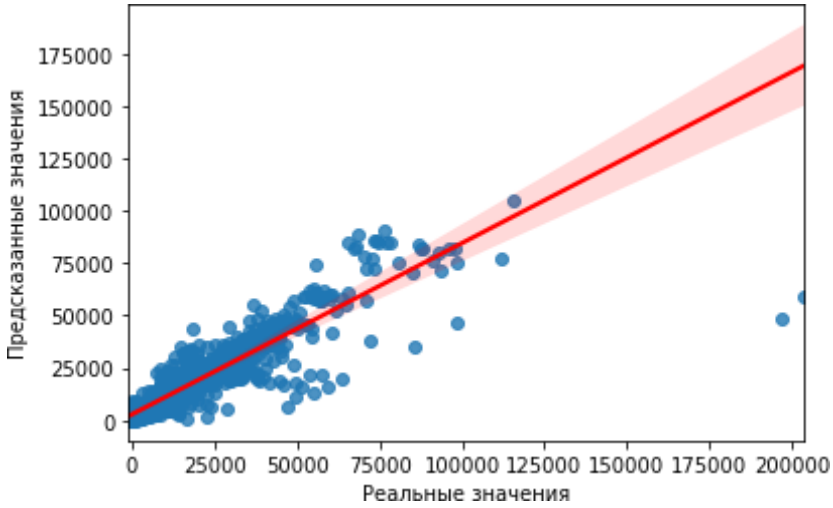
In [53]:
```
%%time
br=BaggingRegressor()
br.fit(X_train, y_train)
y_pred = br.predict(X_test)
rmse_br = np.sqrt(mean_squared_error(y_test, y_pred))
mae_br = mean_absolute_error(y_test, y_pred)
r2_br = r2_score(y_test, y_pred)
print('RMSE:', rmse_br)
print('MAE:', mae_br)
print('R2:', r2_br)
```

```
RMSE: 7602.287900767191
MAE: 3079.6639038123167
R2: 0.8221557084633634
Wall time: 935 ms
```

Значение RMSE имеет достаточно низкий уровень, что говорит о высоком качестве предсказания. Значение MAE (Mean Absolute Error) также достаточно низкое, что указывает на небольшую погрешность предсказания. Значение коэффициента детерминации R2 достаточно высокое и равно 0.82, что говорит о том, что модель предсказывает выручку в достаточной степени.

Точность модели достаточно высокая , модель предсказывает хорошо. Визуализируем полученные результаты :

In [54]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Градиентный бустинг

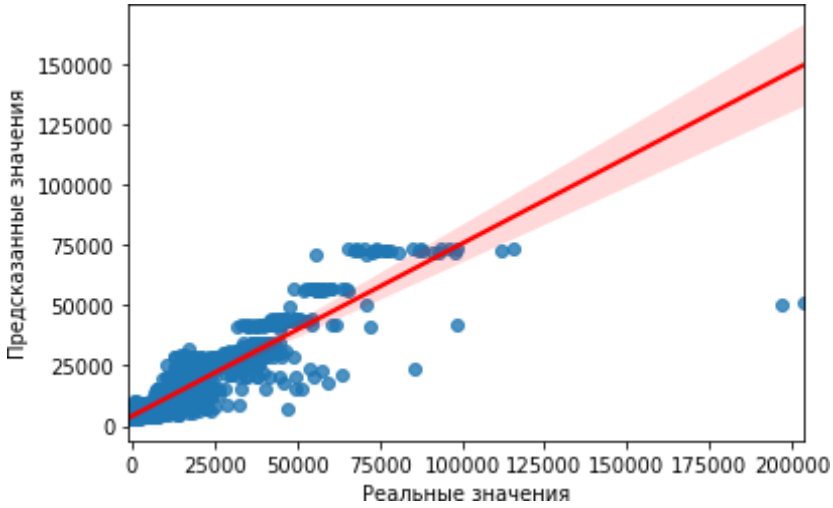
In [55]:
```
%%time
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
y_pred = gbr.predict(X_test)
rmse_gbr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_gbr = mean_absolute_error(y_test, y_pred)
r2_gbr = r2_score(y_test, y_pred)
print('RMSE:', rmse_gbr)
print('MAE:', mae_gbr)
print('R2:', r2_gbr)
```

```
RMSE: 8014.694325362838
MAE: 4062.8166375978285
R2: 0.8023370687272767
Wall time: 2.28 s
```

Значение RMSE (Mean Squared Error) имеет достаточно низкий уровень, что говорит о высоком качестве предсказания. Значение MAE (Mean Absolute Error) также достаточно низкое, что указывает на небольшую погрешность предсказания. Значение коэффициента детерминации R2 достаточно высокое и равно 0.802, что говорит о том, что модель объясняет изменения в целевой переменной в достаточной степени.

Точность модели достаточно высокая , визуализируем на графике результаты:

In [56]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Многослойный перцептрон

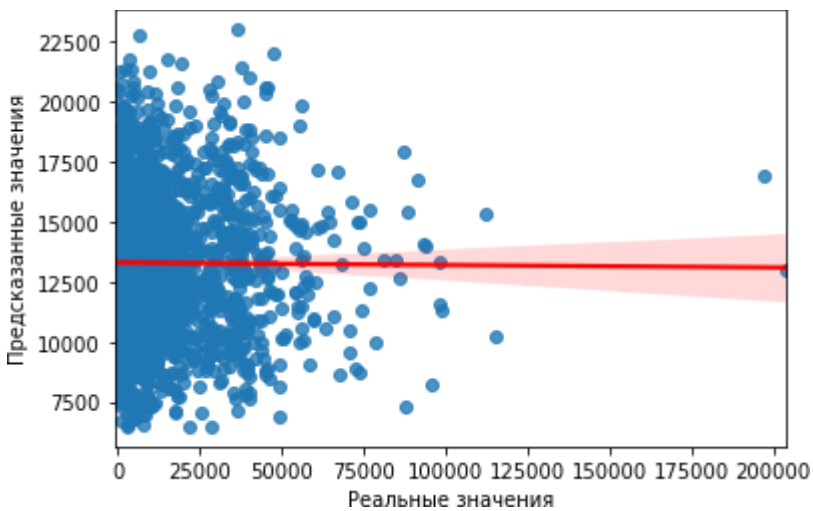
In [57]:
```
%%time
mlp = MLPRegressor()
mlp.fit(X_train, y_train)
y_pred = mlp.predict(X_test)
rmse_mlp = np.sqrt(mean_squared_error(y_test, y_pred))
mae_mlp = mean_absolute_error(y_test, y_pred)
r2_mlp = r2_score(y_test, y_pred)
print('RMSE:', rmse_mlp)
print('MAE:', mae_mlp)
print('R2:', r2_mlp)
```

```
RMSE: 18455.565084206384
MAE: 12696.776084654775
R2: -0.04810791180134433
Wall time: 10.5 s
```

Значение среднеквадратичной ошибки (RMSE) составляет 18455.56. Средняя абсолютная ошибка (MAE) составляет 12696.77. Коэффициент детерминации (R2) составляет -0.048, что свидетельствует о том, что модель не объясняет вариацию в данных и не может быть применима.

Точность модели многослойного перцептрона отрицательная, график выглядит следующим образом

In [58]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Метод ближайших соседей

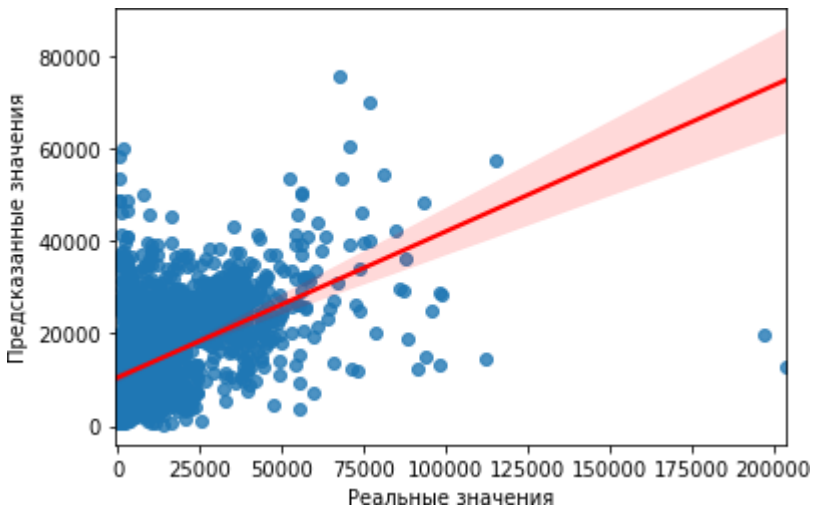
In [59]:
```
%%time
knn = KNeighborsRegressor()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
rmse_knn = np.sqrt(mean_squared_error(y_test, y_pred))
mae_knn = mean_absolute_error(y_test, y_pred)
r2_knn = r2_score(y_test, y_pred)
print('RMSE:', rmse_knn)
print('MAE:', mae_knn)
print('R2:', r2_knn)
```

```
RMSE: 15735.7092946448
MAE: 9773.936837536656
R2: 0.238054423127265
Wall time: 62.6 ms
```

Коэффициент детерминации (R2) составляет 0.238 , что свидетельствует о том, что модель имеет низкую точность и данную модель я не рекомендую использовать для прогнозирования выручки заданного датасета.

Точность модели крайне низкая, график выглядит следующим образом

In [60]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Пассивно-агрессивная регрессия

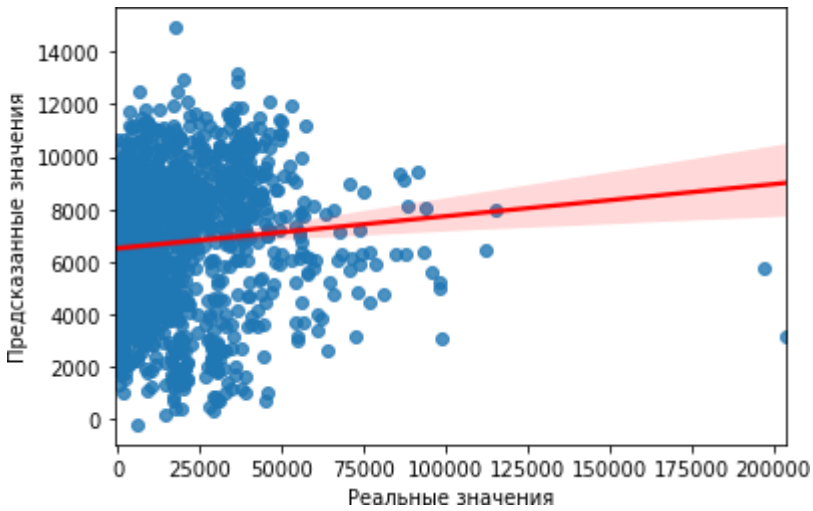
In [61]:
```
par=PassiveAggressiveRegressor()
par.fit(X_train, y_train)
y_pred = par.predict(X_test)
rmse_par = np.sqrt(mean_squared_error(y_test, y_pred))
mae_par = mean_absolute_error(y_test, y_pred)
r2_par = r2_score(y_test, y_pred)
print('RMSE:', rmse_par)
print('MAE:', mae_par)
print('R2:', r2_par)
```

```
RMSE: 20046.82625077322
MAE: 12147.124898486267
R2: -0.2366379141719881
```

Результаты, полученные при применении данного метода в вашем случае, показывают низкую точность модели. Значение коэффициента детерминации R^2 отрицательно, что говорит о том, что модель не подходит для предсказания данных. Значение RMSE и MAE также очень высокие, что означает, что модель допускает большие ошибки при предсказании выходных значений.

Точность модели крайне низкая, график выглядит следующим образом

In [62]:
```
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```



Сравнение моделей:

In [70]:
```
table=pd.DataFrame([['Линейная регрессия',r2, rmse,mae],[ 'Дерево решений ',r2_dtr,rmse_dtr,ma
table
```

Out[70]:

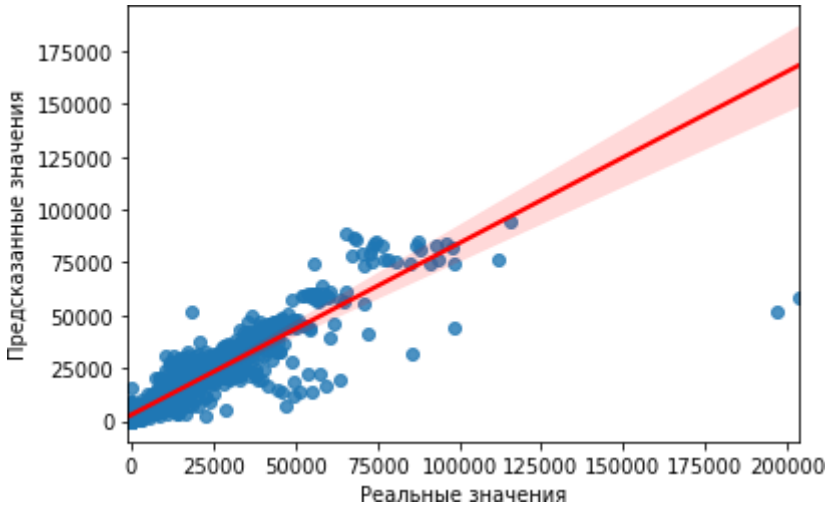
	Модель	R2	RMSE	MAE
0	Линейная регрессия	0.011512	17922.973731	12699.071337
1	Дерево решений	0.725653	9442.221203	3854.066944
2	Случайный лес	0.831741	7581.971640	2857.028243
3	Беггинг	0.822156	7602.287901	3079.663904
4	Градиентный бустинг	0.802337	8014.694325	4062.816638
5	Многослойный перцептрон	-0.048108	18455.565084	12696.776085
6	Метод ближайших соседей	0.238054	15735.709295	12699.071337
7	Пассивно-агрессивная регрессия	-0.236638	20046.826251	12147.124898

Попробуем улучшить модель, которую выбирем для анализа(случайный лес , так как она является наиболее точной)

In [64]:

```
%%time
rfr=RandomForestRegressor(n_estimators=200,
                          max_depth=None,
                          min_samples_split=2,
                          oob_score=True)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

```
RMSE: 7533.508228865771
MAE: 2998.7722501466274
R2: 0.8253591490528268
```



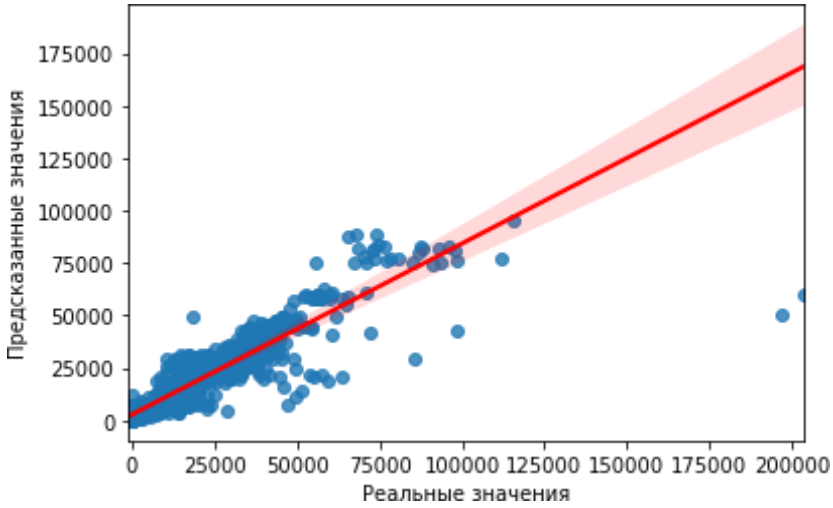

```
Wall time: 18.9 s
```

результат стал немного лучше, но разница небольшая

In [65]:

```
%%time
rfr=RandomForestRegressor(n_estimators=100,
                          max_depth=10,
                          min_samples_split=2,
                          oob_score=True)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

```
RMSE: 7383.770772836312
MAE: 2886.638527416111
R2: 0.8322325450835668
```



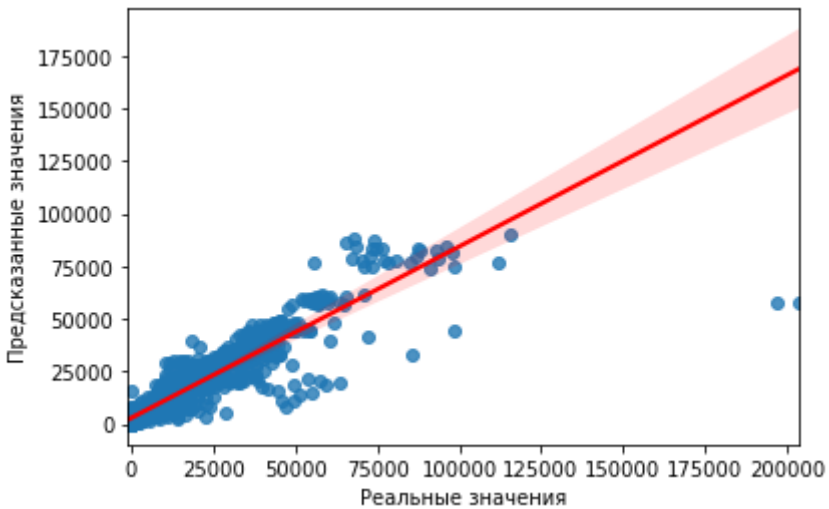

```
Wall time: 6.02 s
```

модель стала предсказывать лучше

In [66]:

```
%%time
rfr=RandomForestRegressor(n_estimators=100,
                          max_depth=20,
                          min_samples_split=5,
                          oob_score=True)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

```
RMSE: 7422.172938053486
MAE: 2990.6784921933336
R2: 0.8304829275972295
```




```
Wall time: 8.75 s
```

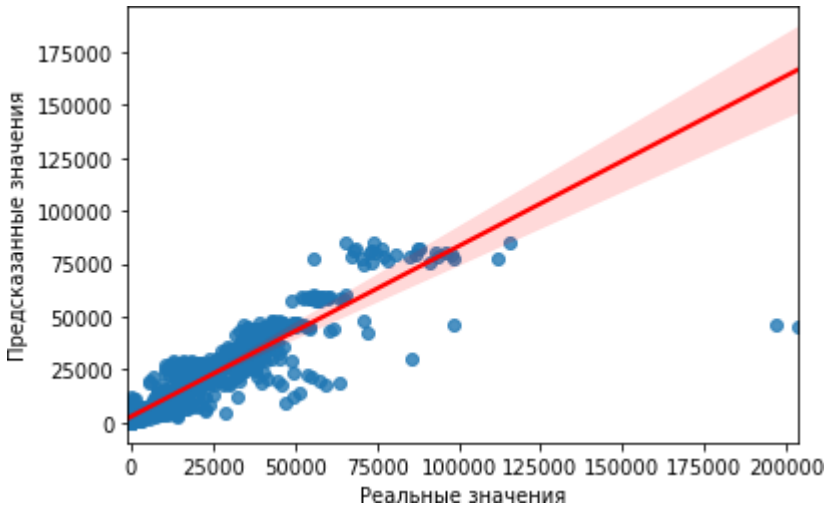
точность чуть хуже предыдущей

In [67]:

```
rfr = RandomForestRegressor(n_estimators=100,
                            max_depth=10,
                            min_samples_split=15,
                            min_samples_leaf=10)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
```

```
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

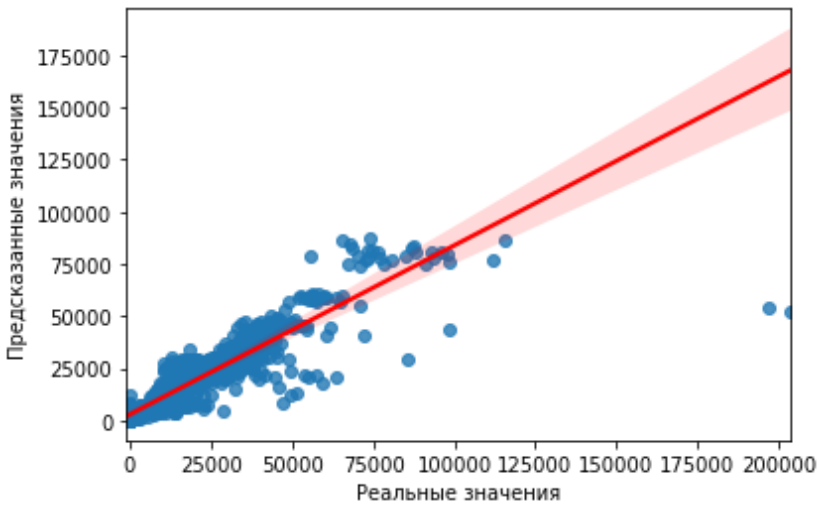
```
RMSE: 7584.475988811381
MAE: 2898.9012248088143
R2: 0.8229880989069607
```



In [68]:

```
%%time
rfr=RandomForestRegressor(n_estimators=100,
                          max_depth=10,
                          min_samples_split=10,
                          min_samples_leaf=3)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred))
mae_rfr = mean_absolute_error(y_test, y_pred)
r2_rfr = r2_score(y_test, y_pred)
print('RMSE:', rmse_rfr)
print('MAE:', mae_rfr)
print('R2:', r2_rfr)
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel('Реальные значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

```
RMSE: 7394.573300401
MAE: 2857.028242564152
R2: 0.8317412951985614
```



```
Wall time: 5.5 s
```

Точность стала немного лучше изначальной

Кросс-валидация

In [69]:

```
scores = cross_val_score(rfr, X_train, y_train, cv=5)
print('Средняя оценка R2 по кросс-валидации:', np.mean(scores))

cv_mse_rfr = -cross_val_score(rfr, X_train, y_train,
                              cv=5,
                              scoring='neg_mean_squared_error')
cv_mae_rfr = -cross_val_score(rfr, X_train, y_train,
                              cv=5,
                              scoring='neg_mean_absolute_error')
cv_rmse_rfr = np.sqrt(cv_mse_rfr)
print('Среднее значение MAE по кросс-валидации:', np.mean(cv_rmse_rfr))
print('Среднее значение RMSE по кросс-валидации:', np.mean(cv_mae_rfr))
```

```
Средняя оценка R2 по кросс-валидации: 0.8773223541166348
Среднее значение MAE по кросс-валидации: 5864.942342696264
Среднее значение RMSE по кросс-валидации: 2799.2285883861
```

После кросс-валидации средняя оценка R2 улучшилась до 0.8782805922519434. Средние значения MAE и RMSE также уменьшились после кросс-валидации до 5857.332005071154 и 2796.952946101086, что указывает на повышение точности модели.

Результат курсовой работы: прогноз выручки

In [361…

```
new_data=pd.DataFrame({'Dept':[1,8,2,4,5,9,13],
                       'IsHoliday':[False,True,True,False,False,False,False],
                       'Stores':[1,7,3,8,1,2,9],
                       'Item_Weight':[9.3,11.89,13.2,19.02,10.7,12.0,7.89],
                       'Item_Fat_Content':[1,2,2,1,2,1,1],
                       'Item_Visibility':[0.016047,0.023401,0.01983,0.03409,0.01282,0.009817,0
                       'Item_MRP':[249.8092,312.023,205.113,402.5412,512.0349,362.838,205.123]
                       'Outlet_Size':[2,2,1,2,3,1,3],
                       'Outlet_Location_Type':[1,3,3,2,1,2,1],
                       'Outlet_Type':[1,3,4,2,4,1,3,]})
```

In [362…

```
new_pred=rfr.predict(new_data)
print(new_pred)
```

```
[21928.51435884 35113.80192862 45464.63772624 36398.7330582
 27731.28936057 31159.39959042 39080.77102325]
```

Мы получили прогнозируемую выручку для данного предприятия

Выводы :

Применение различных моделей машинного обучения позволило получить достаточно точный прогноз выручки на основе данных за предыдущие периоды. Методы, используемые в данной работе, позволяют обрабатывать большой объем данных. В ходе выполнения курсовой работы были изучены и применены различные методы машинного обучения для анализа и прогнозирования выручки предприятия.

В ходе работы было проведено обучение моделей машинного обучения для прогнозирования будущей выручки компании на основе имеющихся данных о прошлой динамике продаж. В

работе использовались следующие методы машинного обучения: линейная регрессия (Linear Regression), дерево решений(Decision Tree), случайный лес (Random Forest), беггинг(Bagging), градиентный бустинг (Gradient Boosting), Многослойный перцептрон(Multilayer Perceptron) и Метод ближайших соседей(KNeighbors).

После тренировки каждой модели мы получаем её точность по показателям R2 score, MAE, MSE ошибок - это позволяет выбирать оптимальную модель для конкретной задачи прогнозирования доходности бизнес-модели предприятия.

Сравнив модели машиного обучения можем сделать вывод, что наиболее точными оказались следующие модели - случайный лес , бэггинг, градиентный бустинг и метод ближайших соседей. Для прогнозирования выручки заданного предприятия использовалась модель случайный лес. В ходе прогнозирования был получен следующий результат :21928.51435884, 35113.80192862, 45464.63772624, 36398.7330582, 27731.28936057, 31159.39959042, 39080.77102325, который показывет выручку предприятия.

Ожидаемый результат работы получен. Данная модель может быть применима для прогнозирования выручки предприятия. Она позволяет финансовым экспертам принимать грамотные и обоснованные решения касательно расходов предприятия на основании полученного прогноза выручки.