Сахарова Е.К. ИУ5-21М, **вариант 14**

Рубежный контроль №1

Тема: Методы обработки данных

- Задача №1 - 14
- Задача №2 - 34

Дополнительные требования по группам:

- Для студентов групп **ИУ5-21М**, ИУ5И-21М, ИУ5Ц-21М - для пары произвольных колонок данных построить график "Диаграмма рассеяния".

Загрузка данных

[ ] ↳ *Скрыто 6 ячеек.*

Задача №14

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "квадратный корень".

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import scipy.stats as stats
```

Этот набор данных содержит подробную информацию о 1000 клиентах и их решении о покупке автомобиля, с учетом их годовой зарплаты, пола и возраста.

```
1 df = pd.read_csv("/content/car_data.csv")
2 df.head()
```

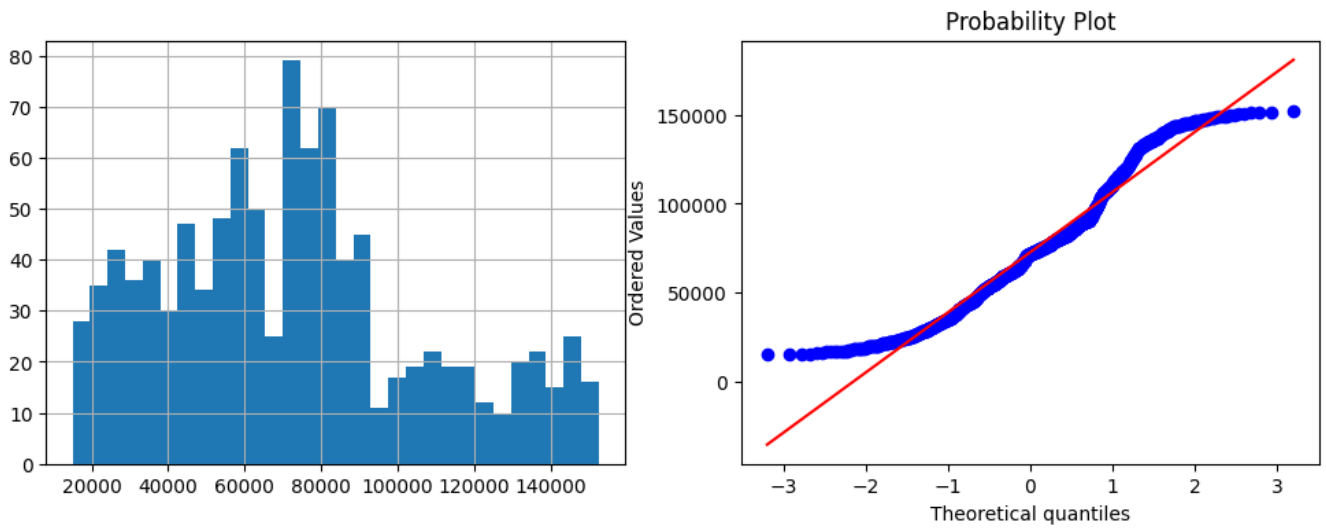
	User ID	Gender	Age	AnnualSalary	Purchased
0	385	Male	35	20000	0
1	681	Male	40	43500	0
2	353	Male	49	74000	0
3	895	Male	40	107500	1
4	661	Male	25	79000	0

```
1 df = pd.get_dummies(df, columns=['Gender'], drop_first=True)
2 df.rename(columns={"Gender_Male": "Gender"}, inplace=True)
3 df.head()
```

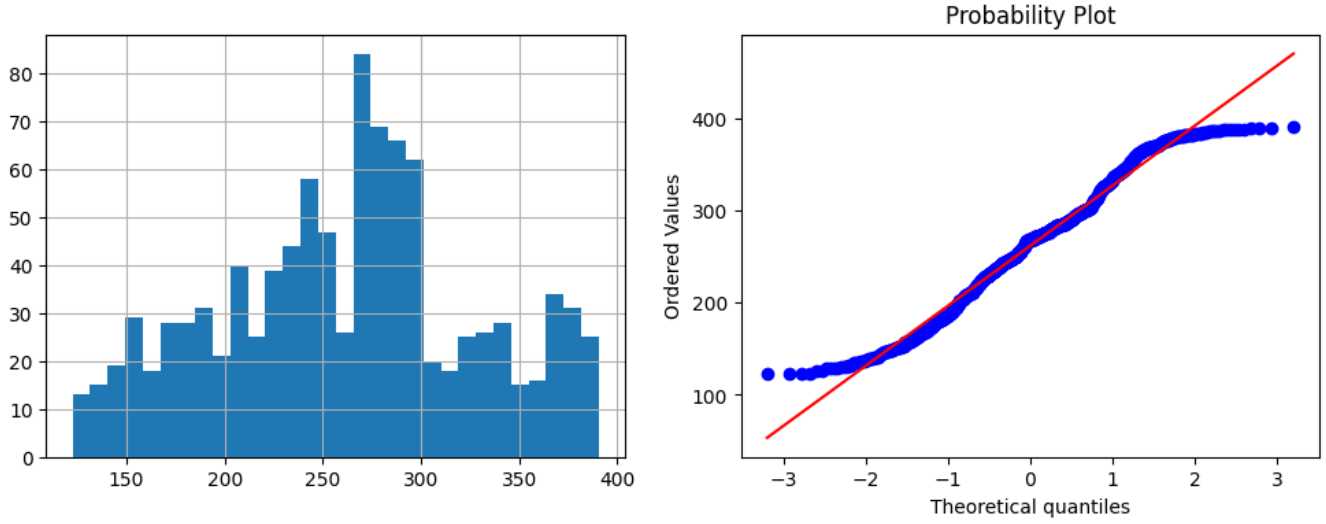
	User ID	Age	AnnualSalary	Purchased	Gender
0	385	35	20000	0	1
1	681	40	43500	0	1
2	353	49	74000	0	1
3	895	40	107500	1	1
4	661	25	79000	0	1

```
1 def diagnostic_plots(df, variable):
2   plt.figure(figsize=(12,4))
3   # гистограмма
4   plt.subplot(1, 2, 1)
5   df[variable].hist(bins=30)
6   ## Q-Q plot
7   plt.subplot(1, 2, 2)
8   stats.probplot(df[variable], dist="norm", plot=plt)
9   plt.show()
```

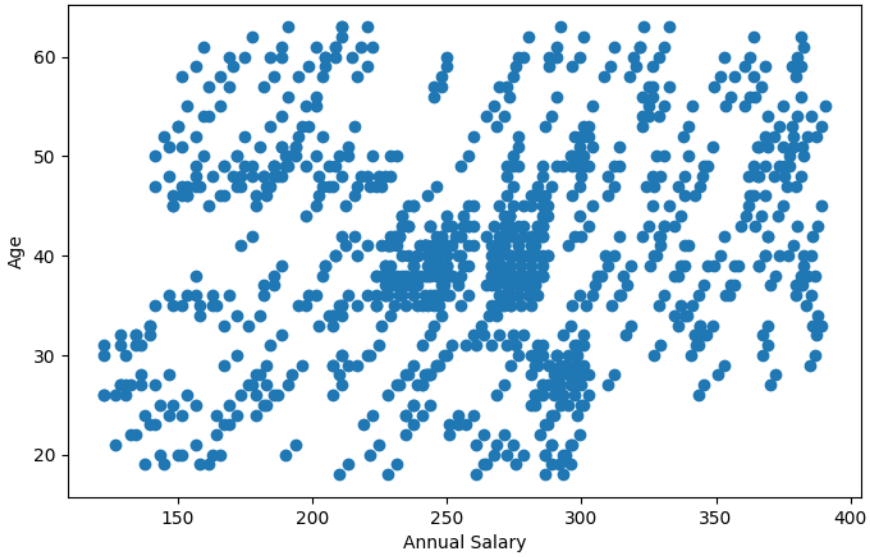
```
1 diagnostic_plots(df, "AnnualSalary")
```




```
1 df["AnnualSalary"] = df["AnnualSalary"] ** (1/2)
2 diagnostic_plots(df, "AnnualSalary")
```




```
1 fig, ax = plt.subplots(figsize=(8, 5))
2 ax.scatter(x = df['AnnualSalary'], y = df['Age'])
3 plt.xlabel("Annual Salary")
4 plt.ylabel("Age");
```



Задача №34

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод вложений (embedded method). Используйте подход на основе линейной или логистической регрессии (в зависимости от того, на решение какой задачи ориентирован выбранный Вами набор данных - задачи регрессии или задачи классификации).

```
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Lasso
from sklearn.svm import LinearSVC
```

```
x = pd.concat([df.iloc[:, 0:3], df.iloc[:, 4]], axis=1)
y = df.iloc[:, 3]
```

```
x.head()
```

	User ID	Age	AnnualSalary	Gender
0	385	35	141.421356	1
1	681	40	208.566536	1
2	353	49	272.029410	1
3	895	40	327.871926	1
4	661	25	281.069386	1

Логистическая регрессия

```
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=500, random_state=1)
e_lr1.fit(x, y)
# Коэффициенты регрессии
e_lr1.coef_
```

```
array([[7.77117661e-05, 2.12434283e-01, 1.54089658e-02, 2.71363098e-01]])
```

```
# Все 4 признака являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(x, y)
sel_e_lr1.get_support()
```

```
array([ True,  True,  True,  True])
```

Линейный классификатор на основе SVM

```
e_lr2 = LinearSVC(C=0.01, penalty="l1", max_iter=2000, dual=False)
e_lr2.fit(x, y)
# Коэффициенты регрессии
e_lr2.coef_
```

```
array([[-0.00027517,  0.03805158,  0.00073502,  0.        ]])
```

```
# последний признак "плохой"
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(x, y)
sel_e_lr2.get_support()
```

```
array([ True,  True,  True, False])
```