Калякин Роман БВТ2203 Вариант 13 ИДЗ№3

Перед началом работы импортируем и установим нужные нам пакеты для анализа данных

In [15]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
import scipy.stats as stat
!pip install statsmodels
import statsmodels.api as sm
!pip install pingouin
import pingouin
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures
```

```
Requirement already satisfied: statsmodels in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (0.14.1)
Requirement already satisfied: numpy<2,>=1.18 in c:\users\tezer\appdata\local\progra
ms\python\python38\lib\site-packages (from statsmodels) (1.23.4)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in c:\users\tezer\appdata\local\pr
ograms\python\python38\lib\site-packages (from statsmodels) (1.9.3)
Requirement already satisfied: pandas!=2.1.0,>=1.0 in c:\users\tezer\appdata\local\p
rograms\python\python38\lib\site-packages (from statsmodels) (2.0.1)
Requirement already satisfied: patsy>=0.5.4 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from statsmodels) (0.5.4)
Requirement already satisfied: packaging>=21.3 in c:\users\tezer\appdata\local\progr
ams\python\python38\lib\site-packages (from statsmodels) (23.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\tezer\appdata\loca
l\programs\python\python38\lib\site-packages (from pandas!=2.1.0,>=1.0->statsmodels)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\tezer\appdata\local\progra
ms\python\python38\lib\site-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2023.
3)
Requirement already satisfied: six in c:\users\tezer\appdata\local\programs\python\p
ython38\lib\site-packages (from patsy>=0.5.4->statsmodels) (1.16.0)
[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: pingouin in c:\users\tezer\appdata\local\programs\pyt
hon\python38\lib\site-packages (0.5.3)
Requirement already satisfied: numpy>=1.19 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from pingouin) (1.23.4)
Requirement already satisfied: scipy>=1.7 in c:\users\tezer\appdata\local\programs\p
ython\python38\lib\site-packages (from pingouin) (1.9.3)
Requirement already satisfied: pandas>=1.0 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from pingouin) (2.0.1)
Requirement already satisfied: matplotlib>=3.0.2 in c:\users\tezer\appdata\local\pro
grams\python\python38\lib\site-packages (from pingouin) (3.7.1)
Requirement already satisfied: seaborn>=0.11 in c:\users\tezer\appdata\local\program
s\python\python38\lib\site-packages (from pingouin) (0.13.0)
Requirement already satisfied: statsmodels>=0.13 in c:\users\tezer\appdata\local\pro
grams\python\python38\lib\site-packages (from pingouin) (0.14.1)
Requirement already satisfied: scikit-learn in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from pingouin) (1.3.2)
Requirement already satisfied: pandas-flavor>=0.2.0 in c:\users\tezer\appdata\local
\programs\python\python38\lib\site-packages (from pingouin) (0.6.0)
Requirement already satisfied: outdated in c:\users\tezer\appdata\local\programs\pyt
hon\python38\lib\site-packages (from pingouin) (0.2.2)
Requirement already satisfied: tabulate in c:\users\tezer\appdata\local\programs\pyt
hon\python38\lib\site-packages (from pingouin) (0.9.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\tezer\appdata\local\prog
rams\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\tezer\appdata\local\pro
grams\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\tezer\appdata\local\pro
grams\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\tezer\appdata\local\progr
ams\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\tezer\appdata\local\program
s\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\tezer\appdata\local\prog
rams\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\tezer\appdata\local
\programs\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin) (2.8.
2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\tezer\appdata
\local\programs\python\python38\lib\site-packages (from matplotlib>=3.0.2->pingouin)
(5.12.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from pandas>=1.0->pingouin) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\tezer\appdata\local\progra
ms\python\python38\lib\site-packages (from pandas>=1.0->pingouin) (2023.3)
Requirement already satisfied: xarray in c:\users\tezer\appdata\local\programs\pytho
n\python38\lib\site-packages (from pandas-flavor>=0.2.0->pingouin) (2023.1.0)
Requirement already satisfied: patsy>=0.5.4 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from statsmodels>=0.13->pingouin) (0.5.4)
Requirement already satisfied: setuptools>=44 in c:\users\tezer\appdata\local\progra
ms\python\python38\lib\site-packages (from outdated->pingouin) (65.5.0)
Requirement already satisfied: littleutils in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from outdated->pingouin) (0.2.2)
Requirement already satisfied: requests in c:\users\tezer\appdata\local\programs\pyt
hon\python38\lib\site-packages (from outdated->pingouin) (2.31.0)
```

```
Requirement already satisfied: joblib>=1.1.1 in c:\users\tezer\appdata\local\program
s\python\python38\lib\site-packages (from scikit-learn->pingouin) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\tezer\appdata\local
\programs\python\python38\lib\site-packages (from scikit-learn->pingouin) (3.2.0)
Requirement already satisfied: zipp>=3.1.0 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from importlib-resources>=3.2.0->matplotlib>=3.
0.2->pingouin) (3.10.0)
Requirement already satisfied: six in c:\users\tezer\appdata\local\programs\python\p
ython38\lib\site-packages (from patsy>=0.5.4->statsmodels>=0.13->pingouin) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\tezer\appdata\lo
cal\programs\python\python38\lib\site-packages (from requests->outdated->pingouin)
(2.1.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\tezer\appdata\local\programs
\python\python38\lib\site-packages (from requests->outdated->pingouin) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\tezer\appdata\local\pr
ograms\python\python38\lib\site-packages (from requests->outdated->pingouin) (1.26.1
2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\tezer\appdata\local\pr
ograms\python\python38\lib\site-packages (from requests->outdated->pingouin) (2022.1
2.7)
[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [40]:
```python
# Вариант 13
# Латинская Америка и страны Карибского бассейна
Y = "Индекс Счастья"
X1,X2,X3 = "ВВП на душу населения", "Свобода", "Щедрость"
our_region = "Latin America and Caribbean"
Z1,Z2 = 2016, 2020

df = pd.read_excel(r"C:\Users\Tezer\Downloads\idz_data.xlsx", sheet_name='2016')
df2 = pd.read_excel(r"C:\Users\Tezer\Downloads\idz_data.xlsx", sheet_name='2020')


df = df[df["Region"]==our_region] # Берём только нужные нам страны
df2 = df2[df2["Regional indicator"]==our_region] # Берём только нужные нам страны
df = df.loc[:, [X1, X2, X3, Y]]
df2 = df2.loc[:, [X1, X2, X3, "Индекс счастья"]]
df.head(5)
#df2.head(5)
```

Out[40]:

	ВВП на душу населения	Свобода	Щедрость	Индекс Счастья
13	1.06879	0.55225	0.22553	7.087
14	1.35943	0.46823	0.22202	7.039
16	1.08754	0.40425	0.15776	6.952
20	1.11508	0.37709	0.11735	6.778
23	1.21670	0.37789	0.31595	6.705

Нормализация Данных

Нормализуем наши данные перед работой с ними

In [41]:
```
for column in df:
    if column != "Country" or column != "Country name":
        mmin = df[column].min()
        mmax = df[column].max()
        df[column] = (df[column]-mmin)/(mmax-mmin)
for column in df2:
    if column != "Country" or column != "Country name":
        mmin = df2[column].min()
        mmax = df2[column].max()
        df2[column] = (df2[column]-mmin)/(mmax-mmin)
df.head(5)
```

Out[41]:

	ВВП на душу населения	Свобода	Щедрость	Индекс Счастья
13	0.714628	1.000000	0.418756	1.000000
14	1.000000	0.805297	0.410726	0.984309
16	0.733038	0.657034	0.263705	0.955868
20	0.760079	0.594095	0.171250	0.898987
23	0.859857	0.595949	0.625629	0.875123

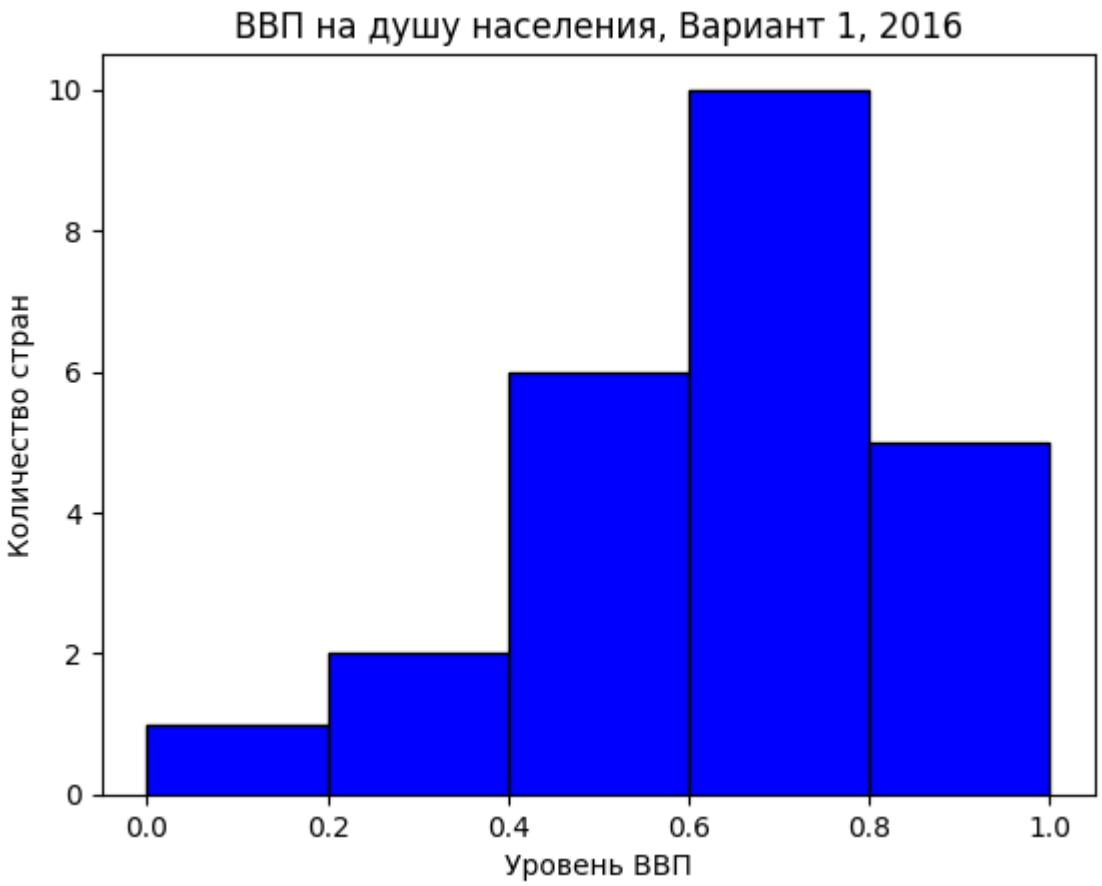
Задание №1

Начнём работать с нашими данными и выведем гистограммы для каждой величины

In [18]:
```
bins = 1 + int(np.log(len(df))/np.log(2))
print(bins)
plt.hist(df[X1], color = 'blue', edgecolor = 'black', bins = bins)

# Add labels
plt.title(f"{X1}, Вариант 1, 2016")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')

scipy.stats.shapiro(df[X1])
# ShapiroResult(statistic=0.9467671513557434, pvalue=0.23034943640232086)
# P-Value > 0.05 => Мы не можем откланить нулевую гипотезу о том, что данные получе
```

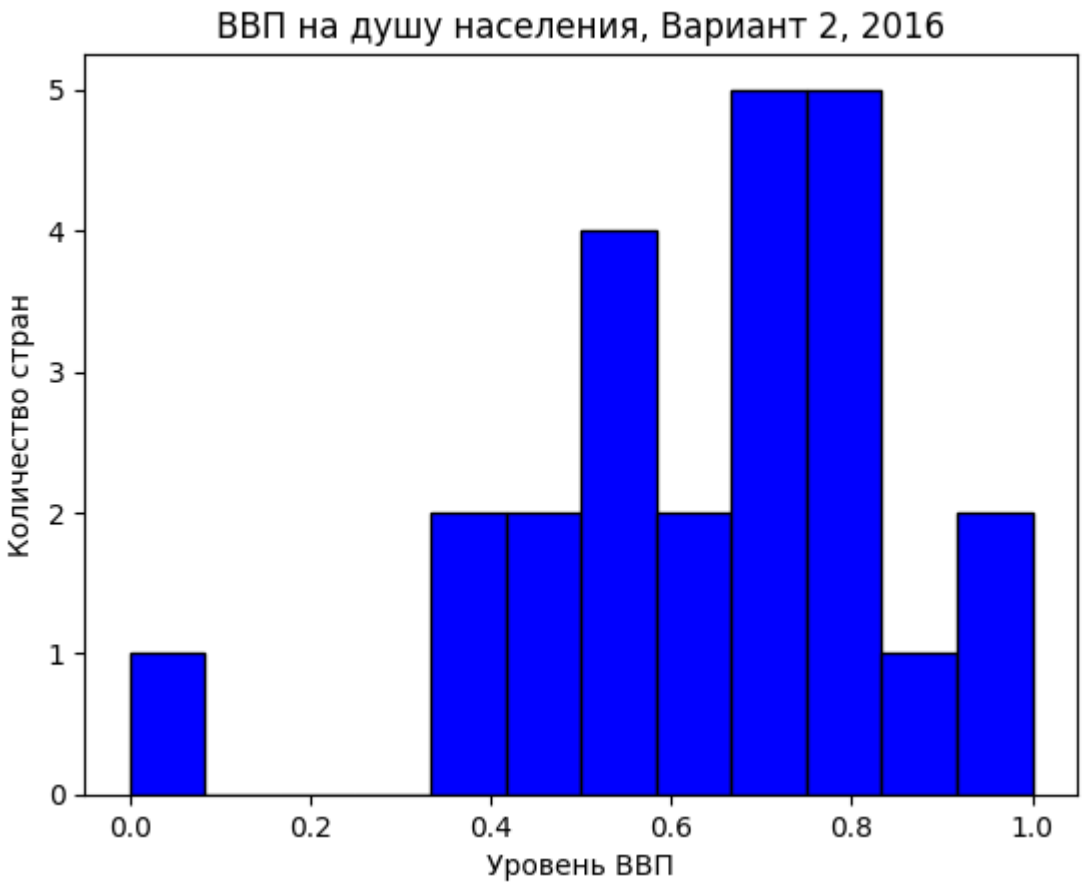
```
5
```

Out[18]:
```
ShapiroResult(statistic=0.9467673897743225, pvalue=0.23035195469856262)
```



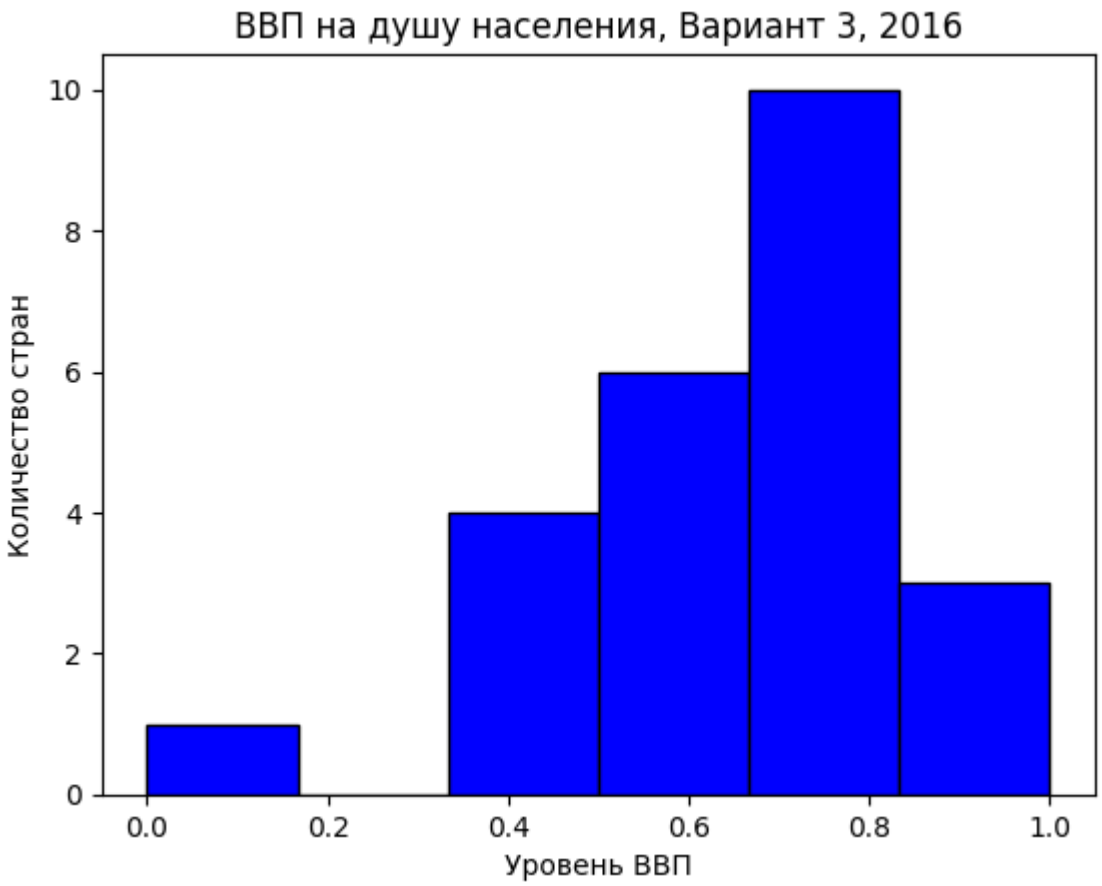
In [29]:
```
# Поставим количество карманов равным 12 и посмотрим на гистограмму
plt.hist(df[X1], color = 'blue', edgecolor = 'black', bins = 12)

# Add labels
plt.title(f"{X1}, Вариант 2, 2016")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

Out[29]:
```
Text(0, 0.5, 'Количество стран')
```



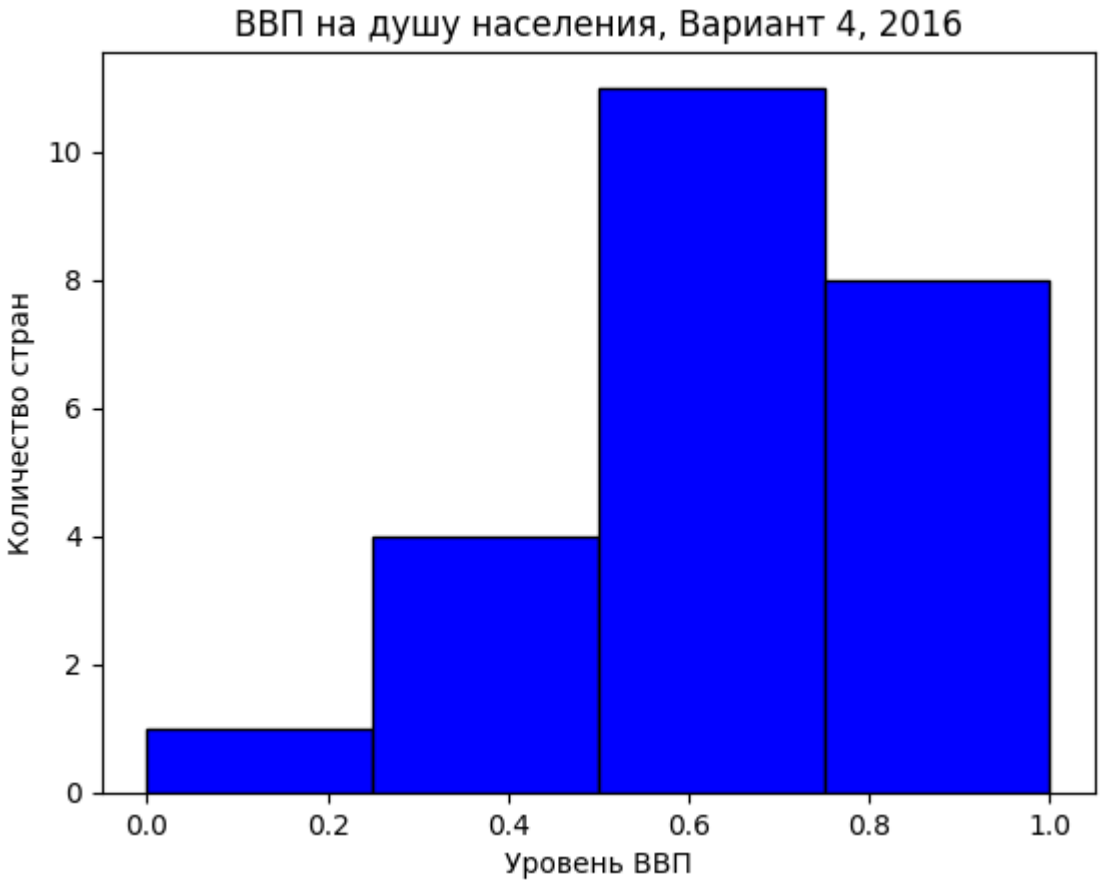
In [30]:
```
# Количество карманов поставим равное 6
plt.hist(df[X1], color = 'blue', edgecolor = 'black', bins = 6)

# Add labels
plt.title(f"{X1}, Вариант 3, 2016")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

Out[30]:
```
Text(0, 0.5, 'Количество стран')
```



In [31]:
```
# Количество карманов поставим равное 4
plt.hist(df[X1], color = 'blue', edgecolor = 'black', bins = 4)

# Add labels
plt.title(f"{X1}, Вариант 4, 2016")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

Out[31]:
```
Text(0, 0.5, 'Количество стран')
```



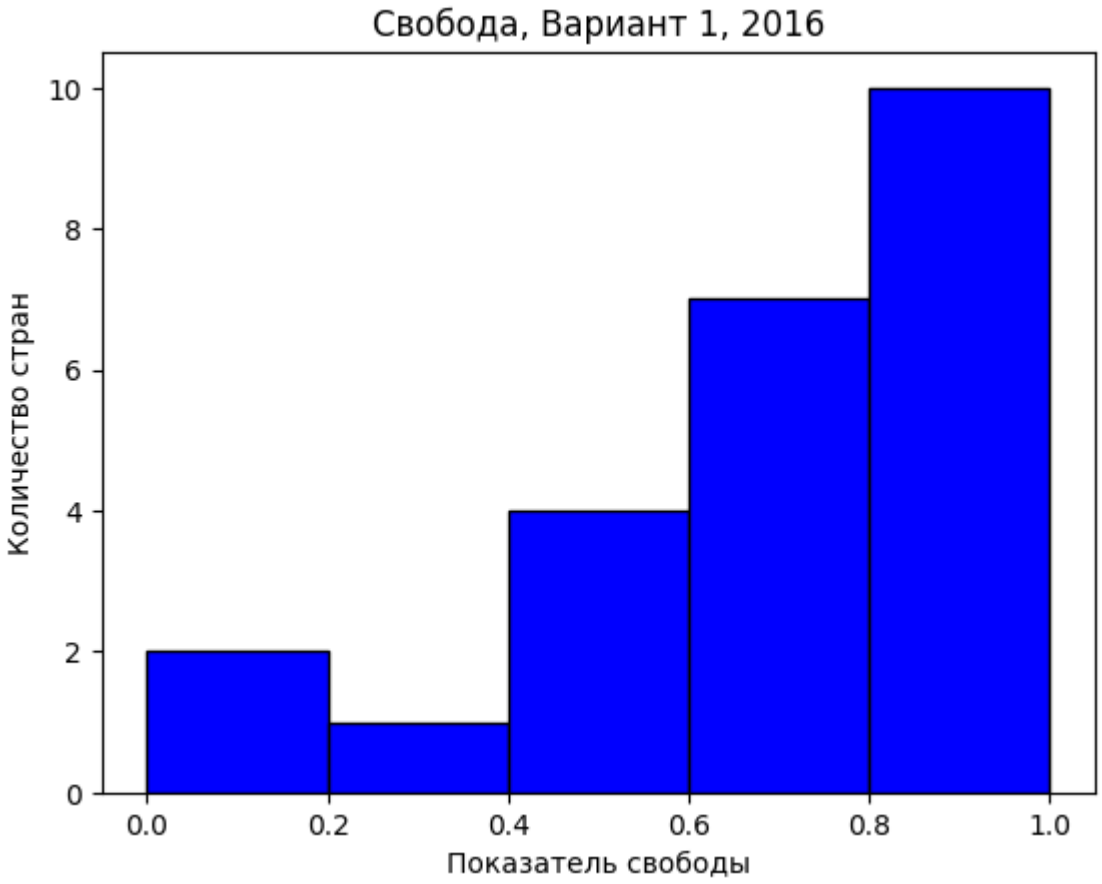
Как мы можем видеть на гистограммах, лучше всего подходит первый варинат, где у нас количество карманов равно 5. Вид гистограммы, а так же тест Шапиро-Уилка дают нам право предположить, что данные распределены по закону нормального распределения.

Рассмотрим так же остальные параметры и сделаем вывод о природе их распределения.

In [32]:
```
bins = 1 + int(np.log(len(df[X2]))/np.log(2))
print(bins)
plt.hist(df[X2], color = 'blue', edgecolor = 'black', bins = bins)

# Add labels
plt.title(f"{X2}, Вариант 1, 2016")
plt.xlabel('Показатель свободы')
plt.ylabel('Количество стран')
scipy.stats.shapiro(df[X2])
```

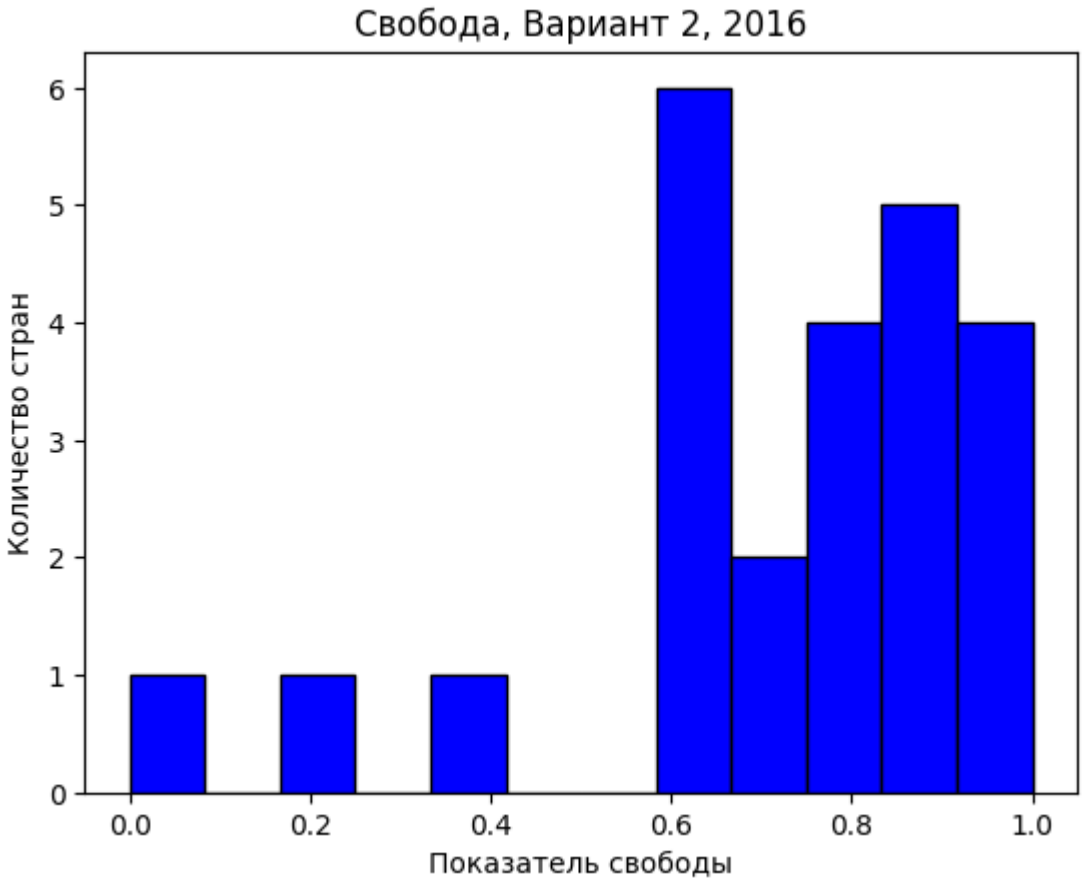
```
5
```

Out[32]:
```
ShapiroResult(statistic=0.8662621378898621, pvalue=0.00444824481382966)
```



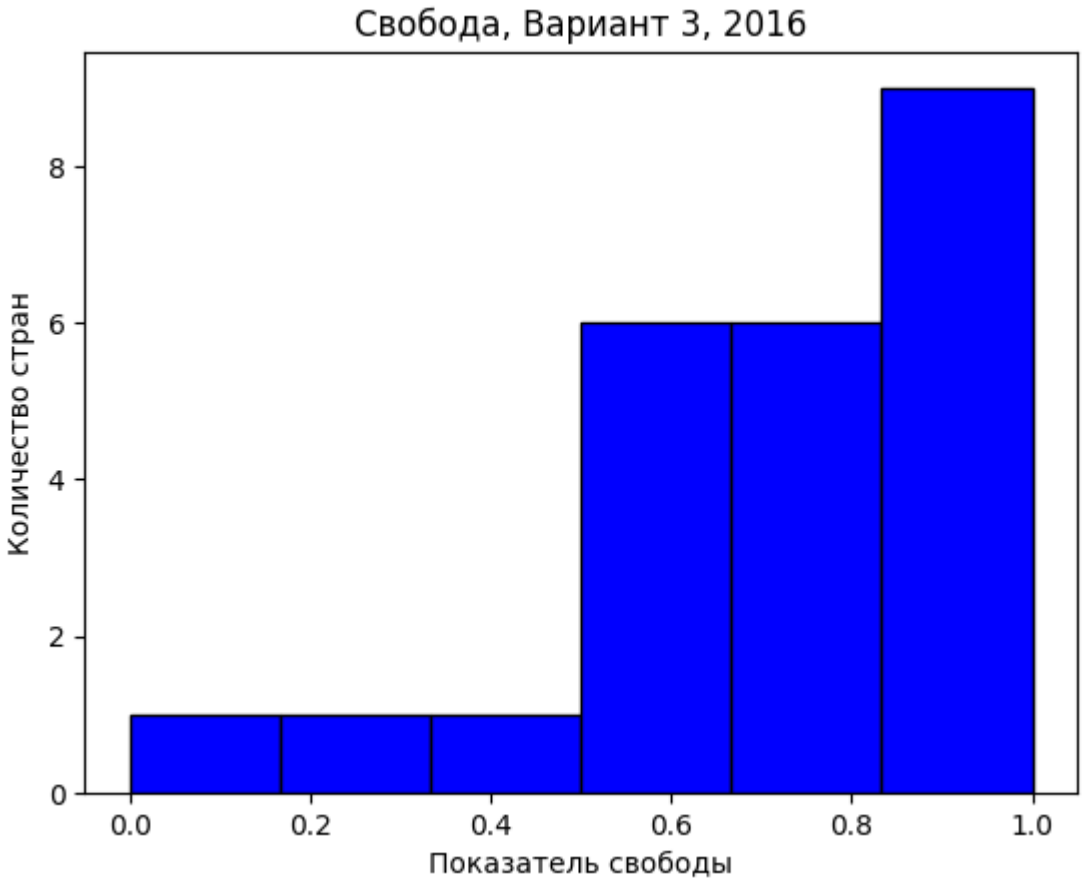
In [33]:
```
plt.hist(df[X2], color = 'blue', edgecolor = 'black', bins = 12)

# Add labels
plt.title(f"{X2}, Вариант 2, 2016")
plt.xlabel('Показатель свободы')
plt.ylabel('Количество стран')
```

Out[33]:
```
Text(0, 0.5, 'Количество стран')
```



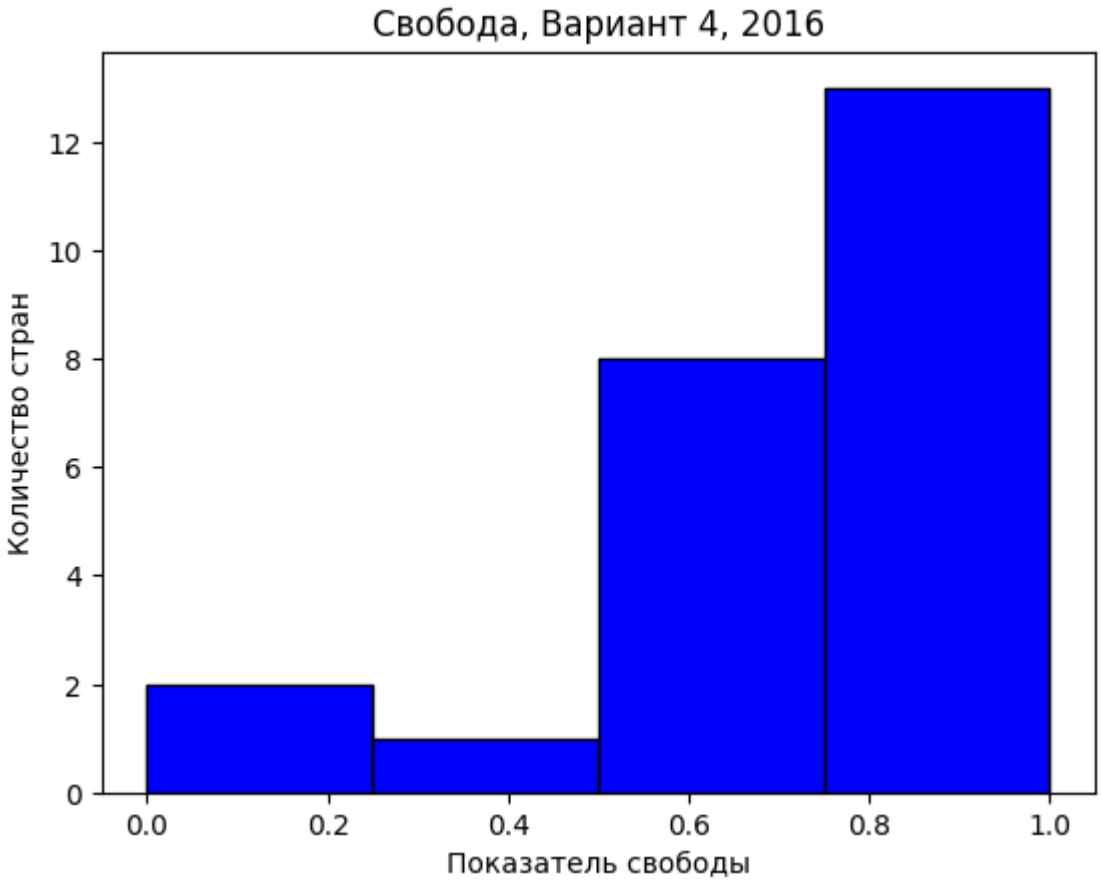
In [34]:
```
plt.hist(df[X2], color = 'blue', edgecolor = 'black', bins = 6)

# Add labels
plt.title(f"{X2}, Вариант 3, 2016")
plt.xlabel('Показатель свободы')
plt.ylabel('Количество стран')
```

Out[34]:
```
Text(0, 0.5, 'Количество стран')
```



In [35]:
```
plt.hist(df[X2], color = 'blue', edgecolor = 'black', bins = 4)

# Add labels
plt.title(f"{X2}, Вариант 4, 2016")
plt.xlabel('Показатель свободы')
plt.ylabel('Количество стран')
```

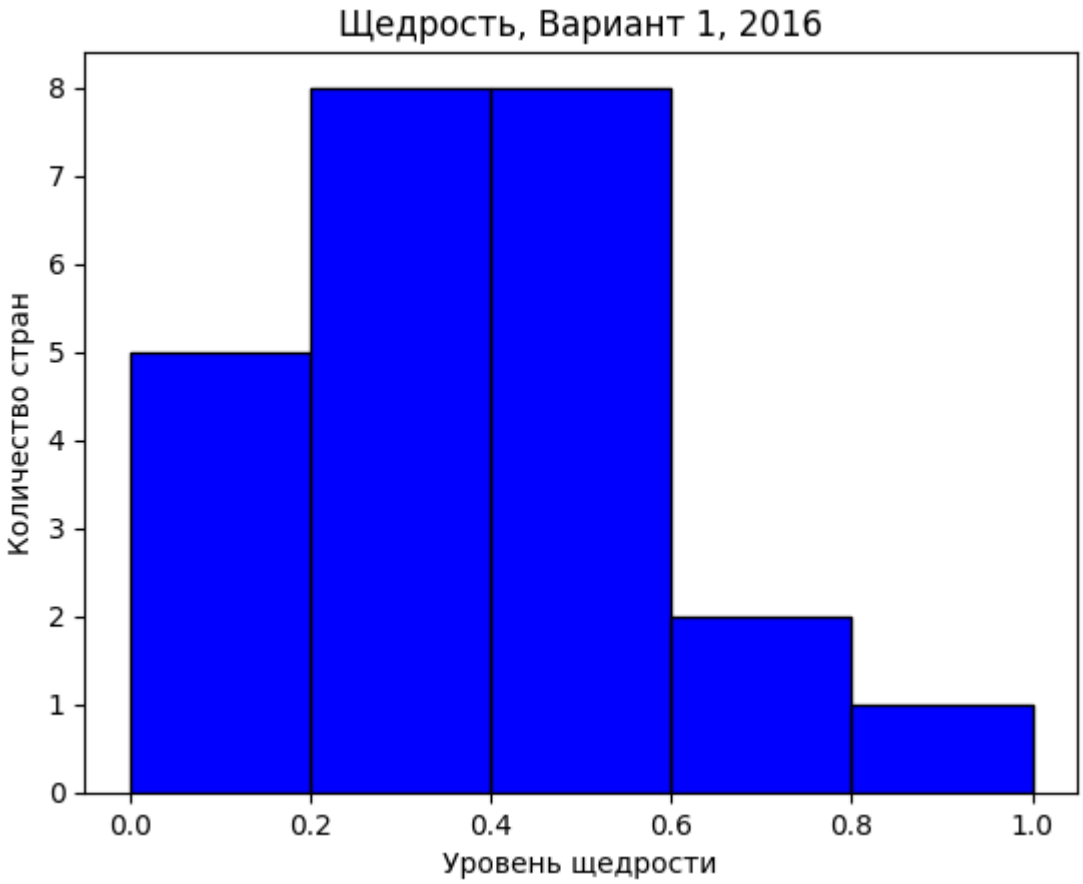
Out[35]:
```
Text(0, 0.5, 'Количество стран')
```



Как мы можем увидеть, распределение напоминает экспоненциальное. Лучшее количество карманов = 6. Тест Шапиро-Уилка так же говорит о том, что это точно не нормальное распределение.

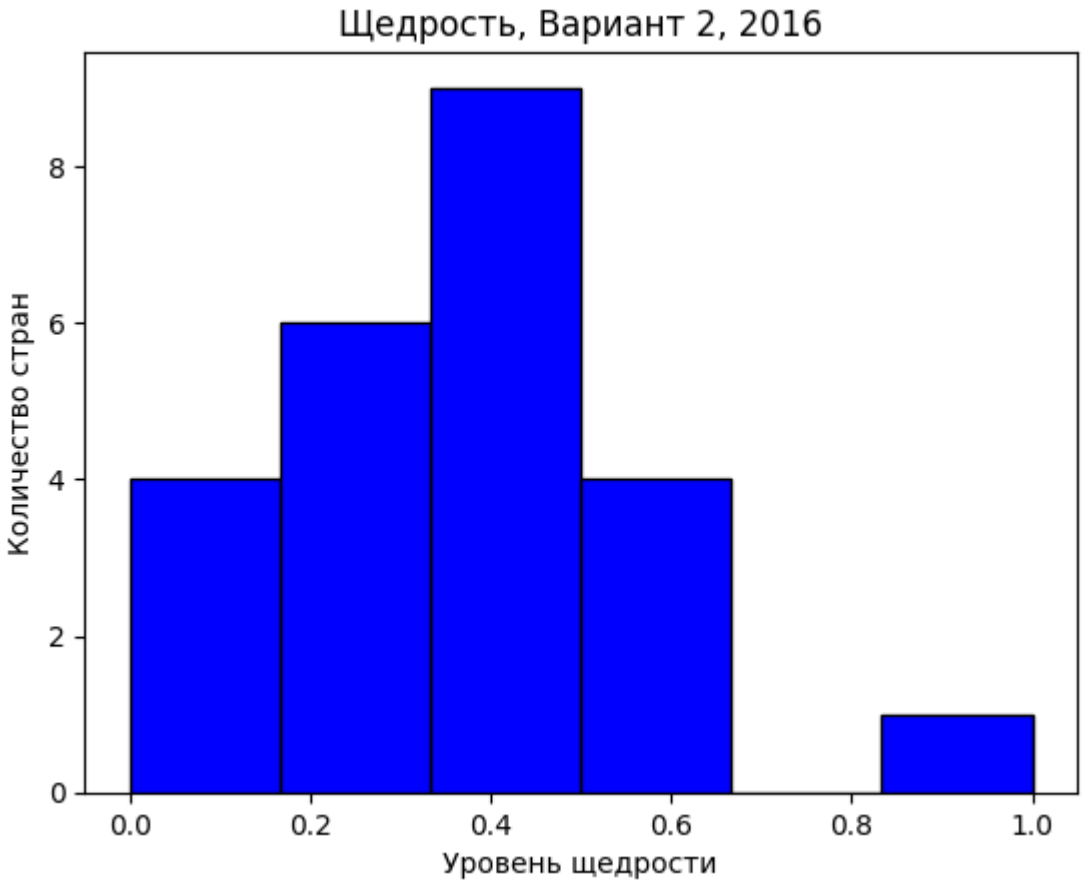
In [36]:
```
bins = 1 + int(np.log(len(df[X2]))/np.log(2))
plt.hist(df[X3], color = 'blue', edgecolor = 'black', bins = bins)

plt.title(f"{X3}, Вариант 1, 2016")
plt.xlabel('Уровень щедрости')
plt.ylabel('Количество стран')
scipy.stats.shapiro(df[X3])
```

Out[36]:
```
ShapiroResult(statistic=0.9518381357192993, pvalue=0.29681602120399475)
```



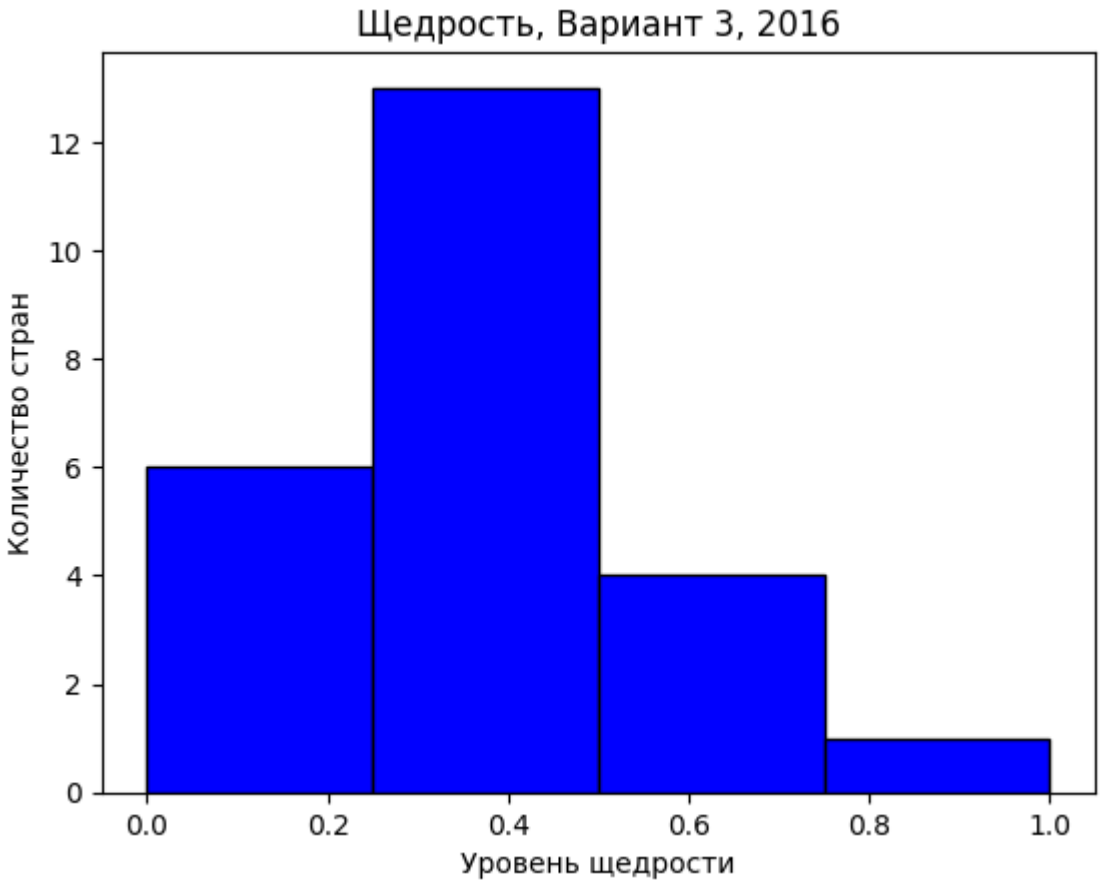
In [37]:
```
plt.hist(df[X3], color = 'blue', edgecolor = 'black', bins = 6)

plt.title(f"{X3}, Вариант 2, 2016")
plt.xlabel('Уровень щедрости')
plt.ylabel('Количество стран')
```

Out[37]:
```
Text(0, 0.5, 'Количество стран')
```



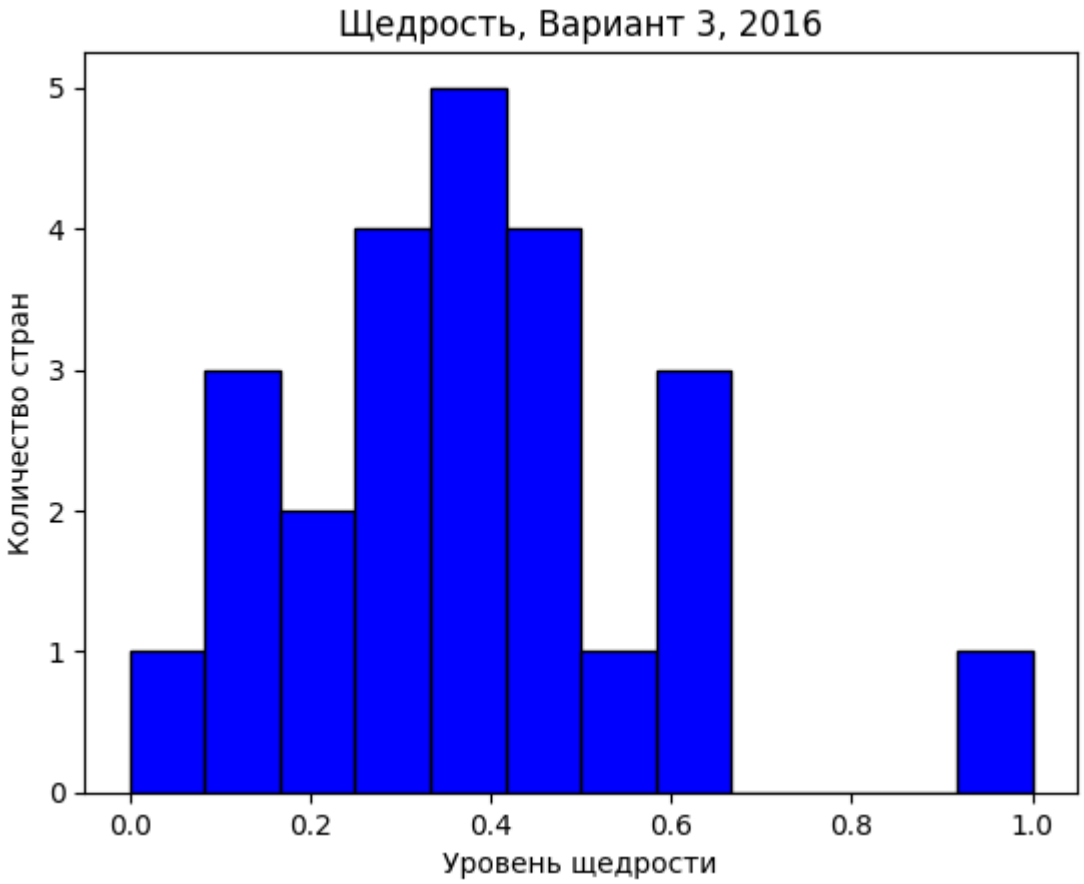
In [38]:
```
plt.hist(df[X3], color = 'blue', edgecolor = 'black', bins = 4)

plt.title(f"{X3}, Вариант 3, 2016")
plt.xlabel('Уровень щедрости')
plt.ylabel('Количество стран')
```

Out[38]:
```
Text(0, 0.5, 'Количество стран')
```



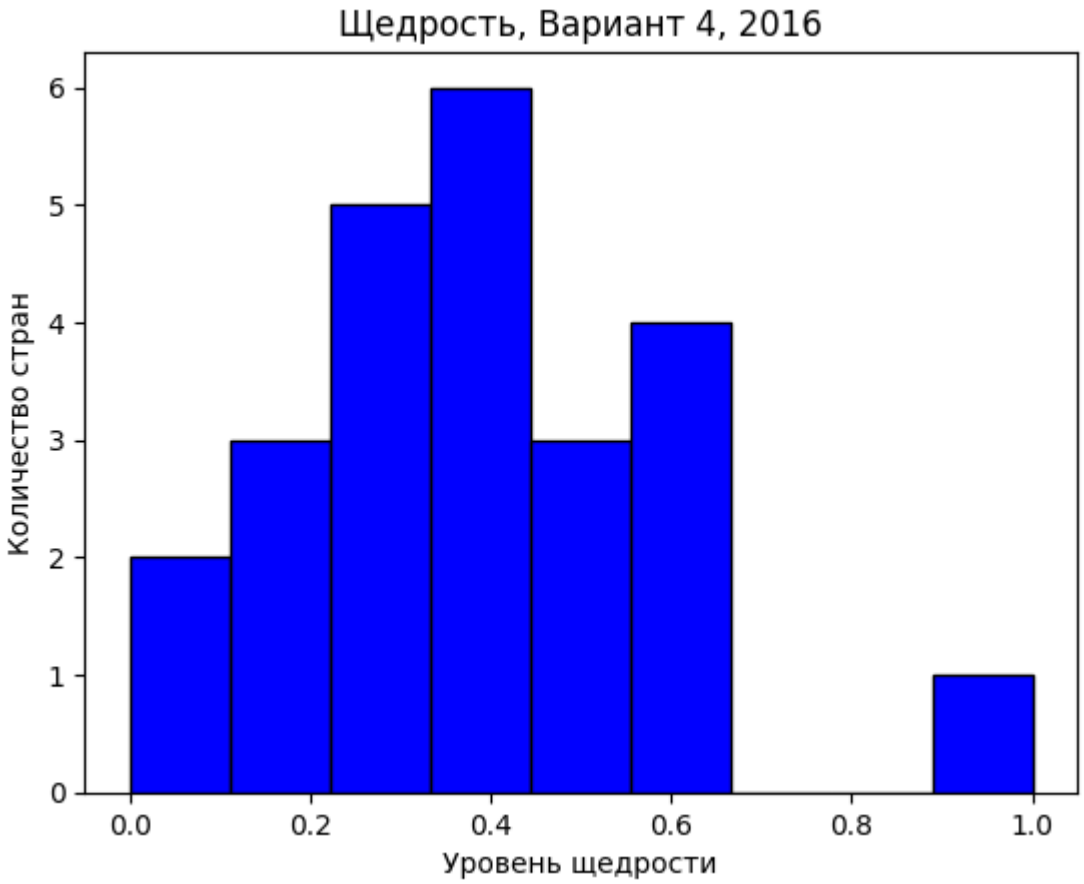
In [39]:
```
plt.hist(df[X3], color = 'blue', edgecolor = 'black', bins = 12)

plt.title(f"{X3}, Вариант 3, 2016")
plt.xlabel('Уровень щедрости')
plt.ylabel('Количество стран')
```

Out[39]:
```
Text(0, 0.5, 'Количество стран')
```



In [40]:
```
plt.hist(df[X3], color = 'blue', edgecolor = 'black', bins = 9)

plt.title(f"{X3}, Вариант 4, 2016")
plt.xlabel('Уровень щедрости')
plt.ylabel('Количество стран')
```

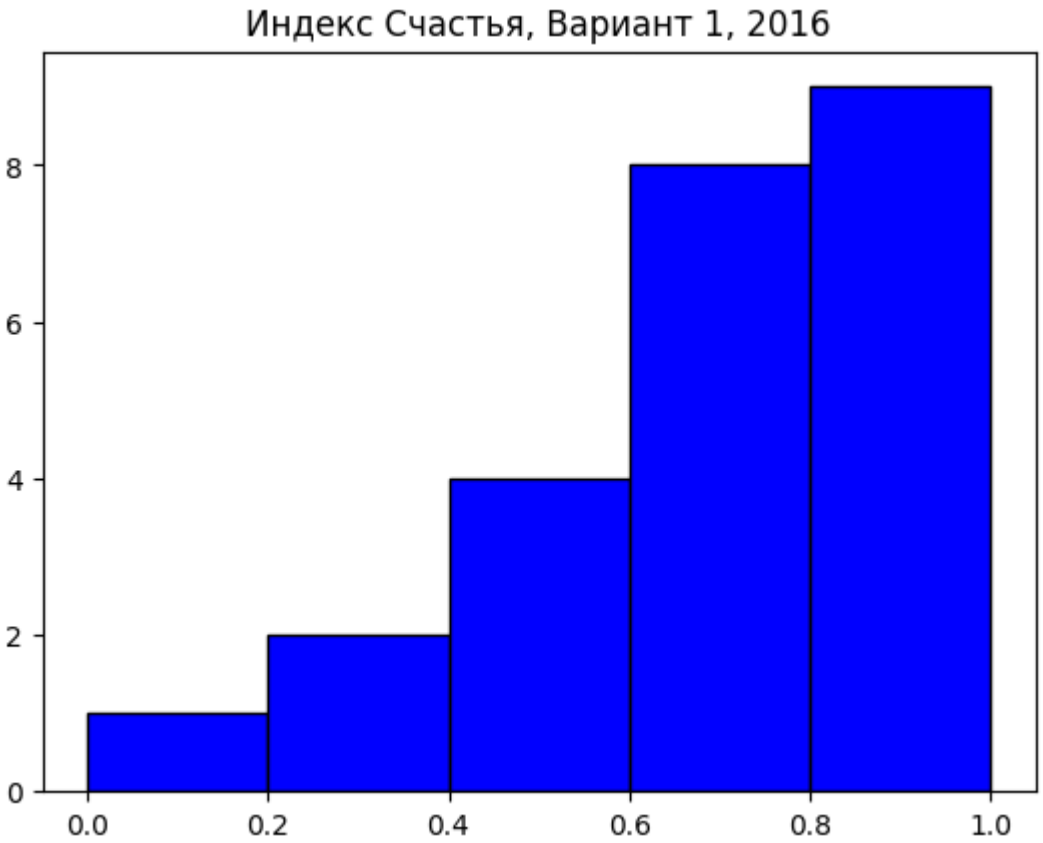
Out[40]:
```
Text(0, 0.5, 'Количество стран')
```



Распределение напоминает нормальное. Тест Шапиро-Уилка этому так же не противоречит.

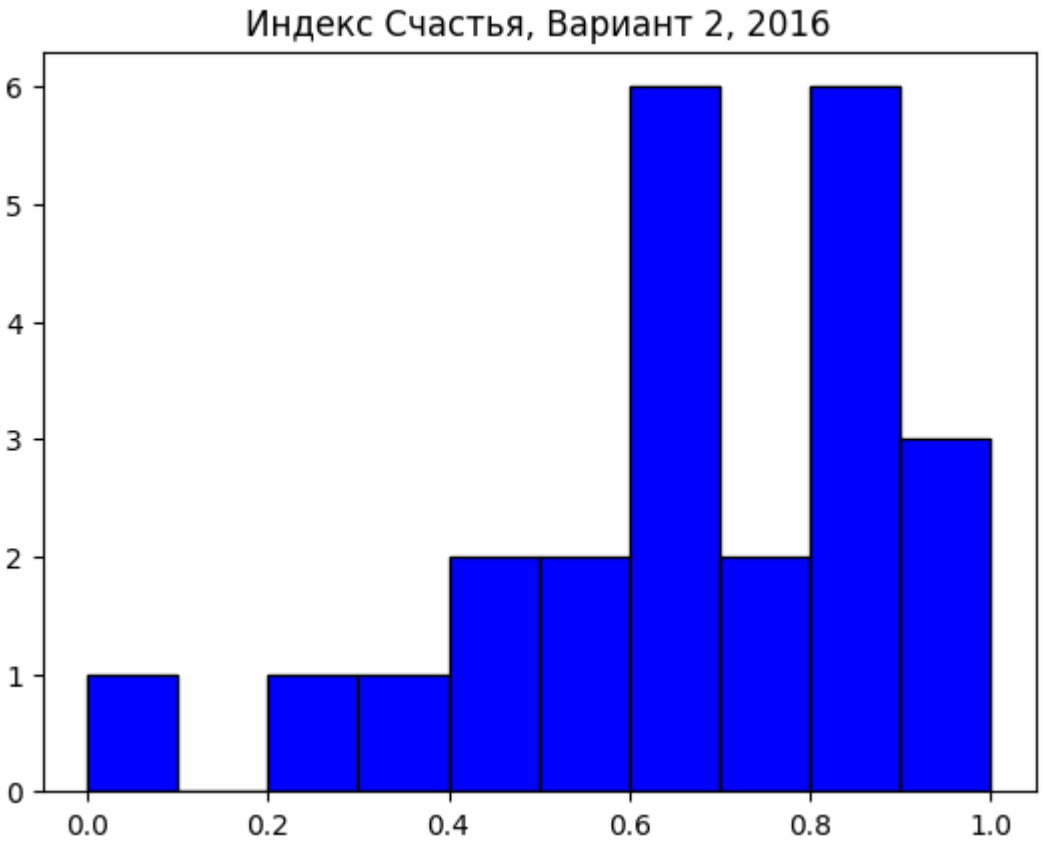
In [41]:
```
bins = 1 + int(np.log(len(df[X2]))/np.log(2))
plt.hist(df[Y], color = 'blue', edgecolor = 'black', bins = bins)

plt.title(f"{Y}, Вариант 1, 2016")
scipy.stats.shapiro(df[Y])
```

Out[41]:
```
ShapiroResult(statistic=0.9314134120941162, pvalue=0.10484544187784195)
```



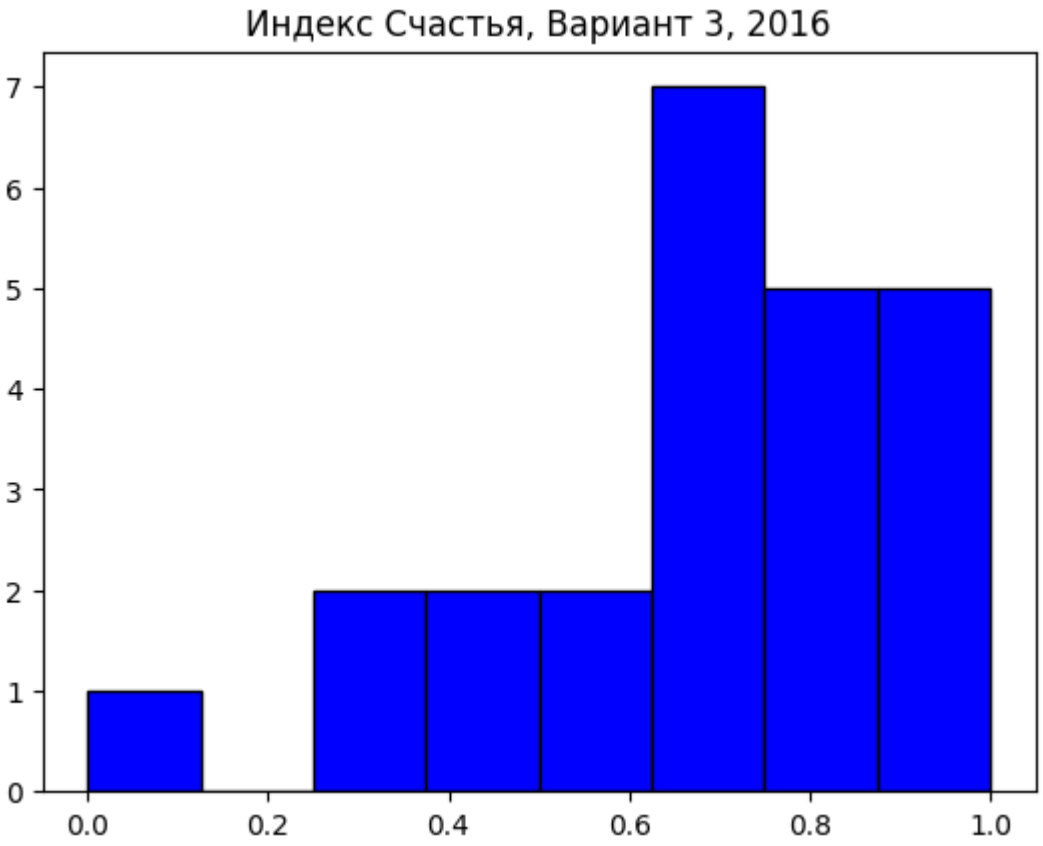
In [42]:
```
plt.hist(df[Y], color = 'blue', edgecolor = 'black', bins = 10)

plt.title(f"{Y}, Вариант 2, 2016")
```

Out[42]:
```
Text(0.5, 1.0, 'Индекс Счастья, Вариант 2, 2016')
```



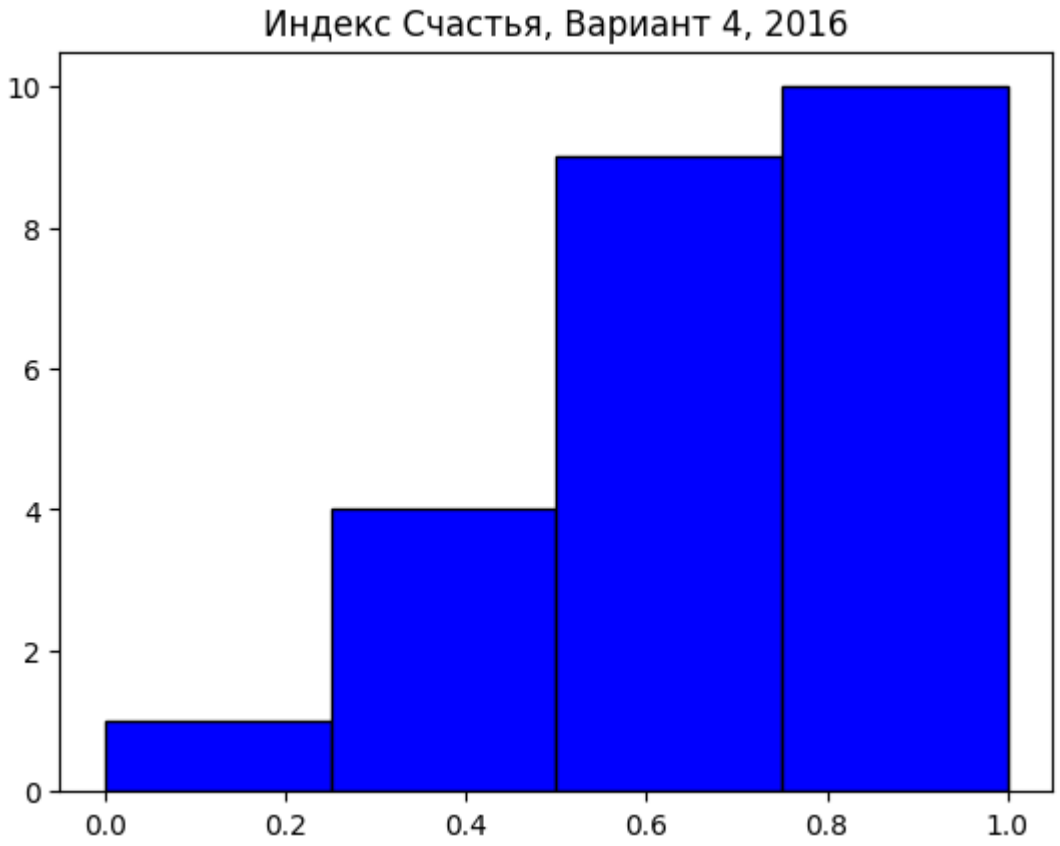
In [43]:
```
plt.hist(df[Y], color = 'blue', edgecolor = 'black', bins = 8)

plt.title(f"{Y}, Вариант 3, 2016")
```

Out[43]:
```
Text(0.5, 1.0, 'Индекс Счастья, Вариант 3, 2016')
```



In [44]:
```
plt.hist(df[Y], color = 'blue', edgecolor = 'black', bins = 4)

plt.title(f"{Y}, Вариант 4, 2016")
```

Out[44]:
```
Text(0.5, 1.0, 'Индекс Счастья, Вариант 4, 2016')
```

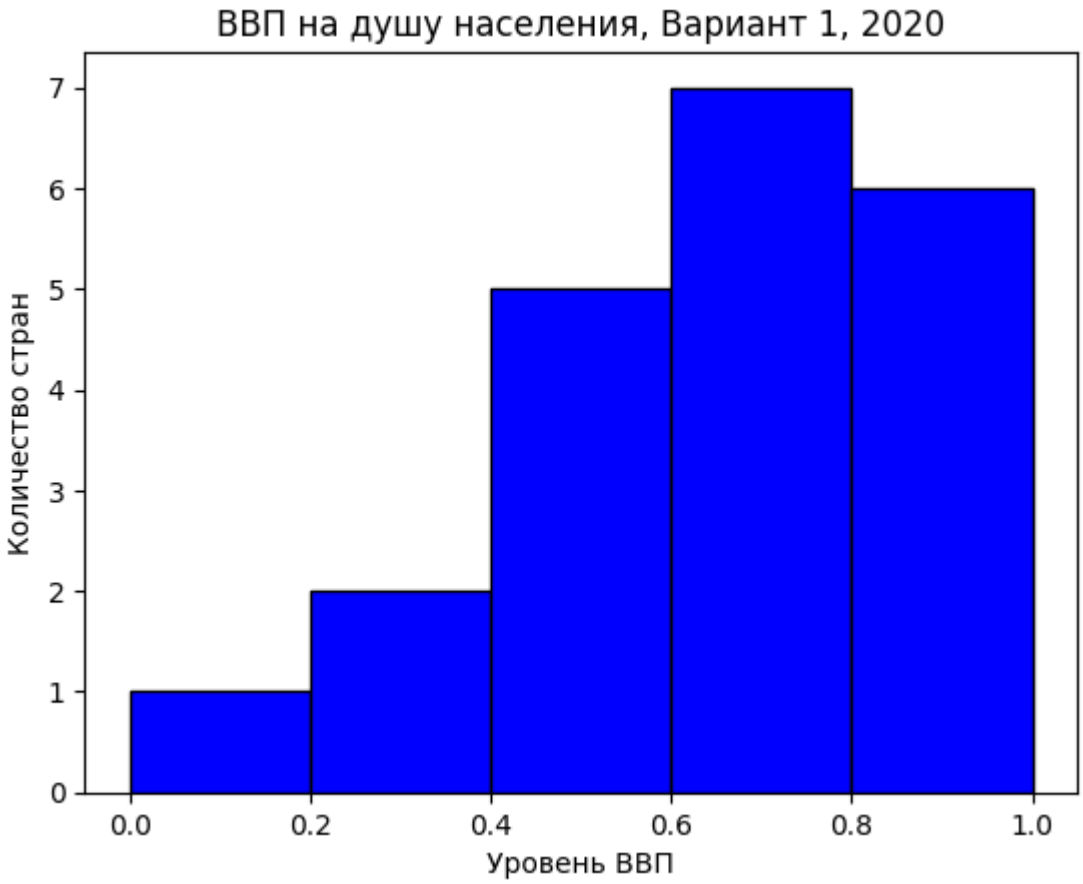


При количестве карманов, равному 5, распределение очень похоже на экспенциальное. Остальные его так же напоминают, но в меньшей степени.

In [88]:
```
bins = 1 + int(np.log(len(df2))/np.log(2))
print(bins)
plt.hist(df2[X1], color = 'blue', edgecolor = 'black', bins = bins)

# Add labels
plt.title(f"{X1}, Вариант 1, 2020")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')

scipy.stats.shapiro(df2[X1])
```

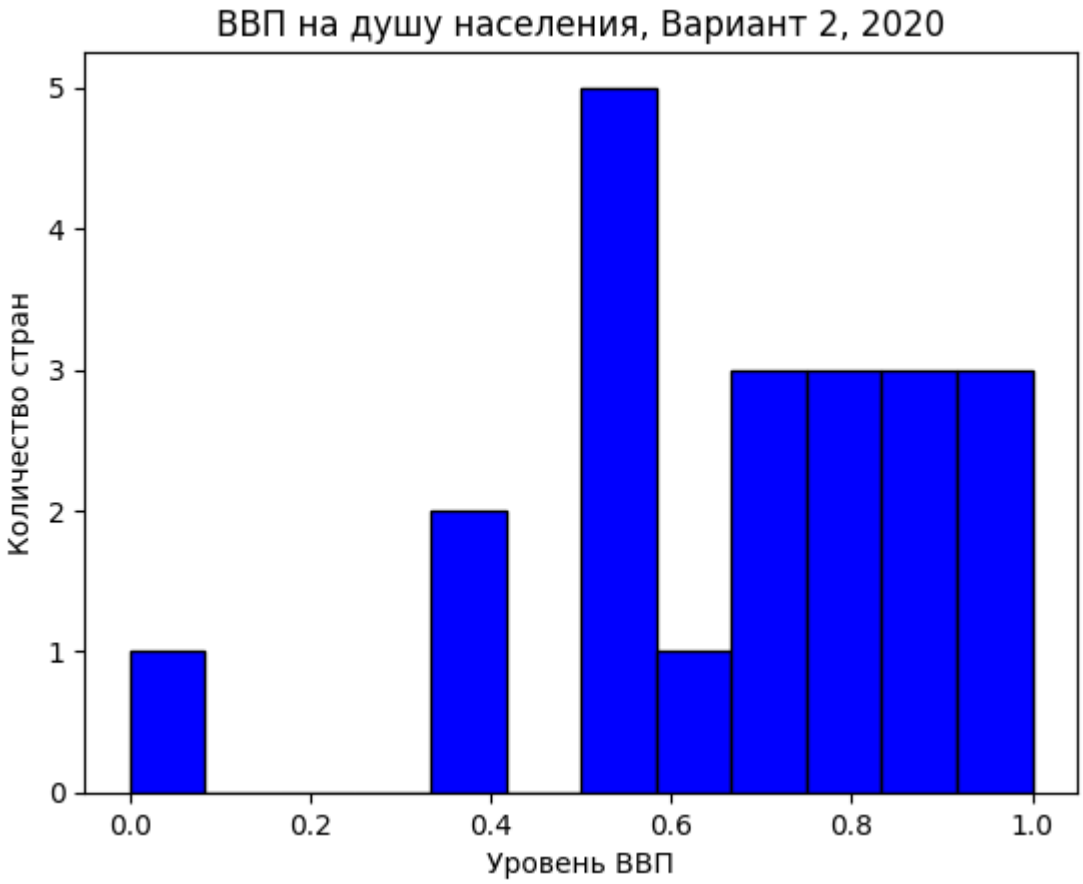
```
5
```

Out[88]:
```
ShapiroResult(statistic=0.9281354546546936, pvalue=0.12629683315753937)
```



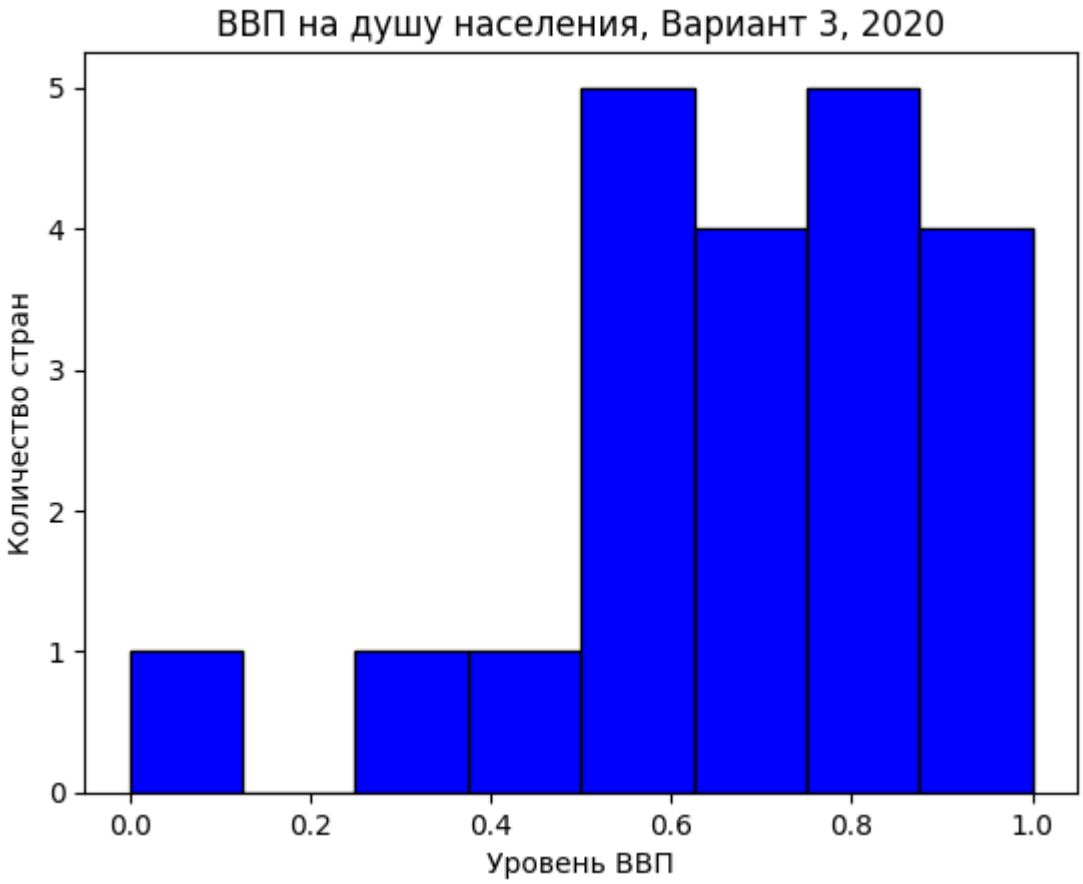
In [89]:
```
plt.hist(df2[X1], color = 'blue', edgecolor = 'black', bins = 12)

plt.title(f"{X1}, Вариант 2, 2020")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

Out[89]:
```
Text(0, 0.5, 'Количество стран')
```



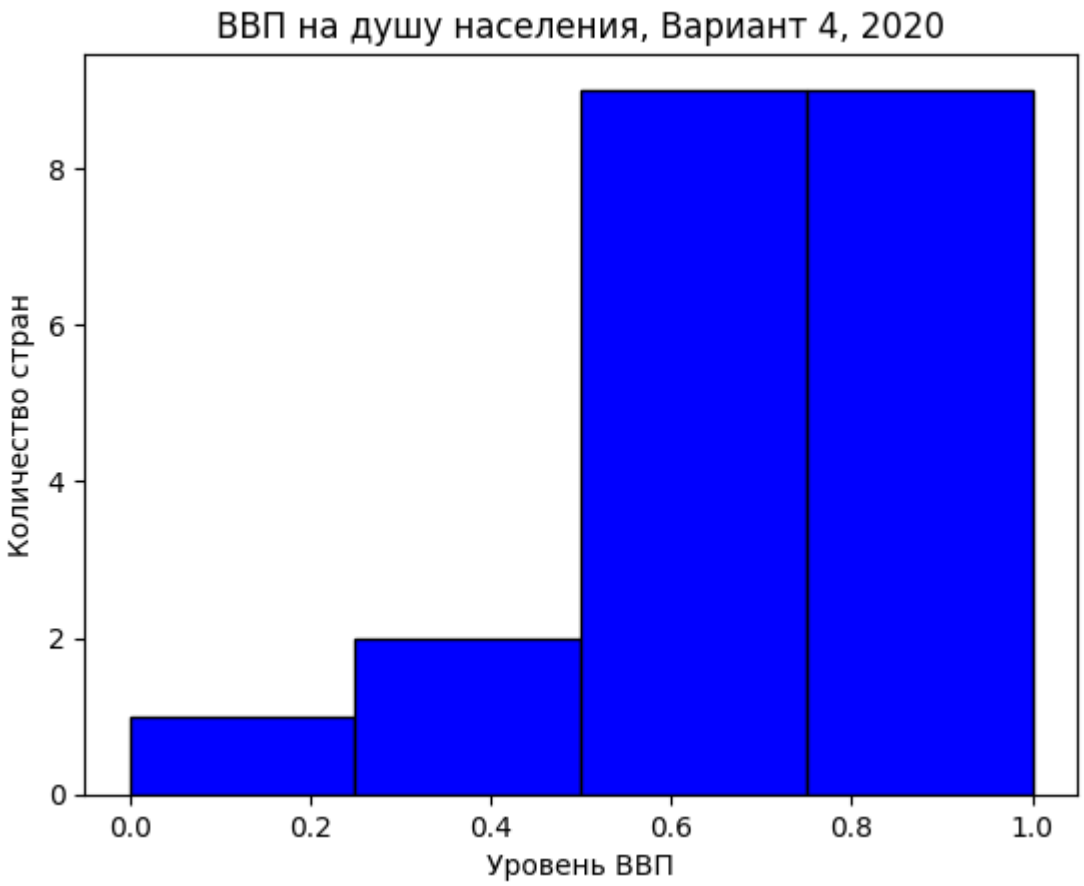
In [90]:
```
plt.hist(df2[X1], color = 'blue', edgecolor = 'black', bins = 8)

# Add labels
plt.title(f"{X1}, Вариант 3, 2020")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

Out[90]:
```
Text(0, 0.5, 'Количество стран')
```



In [92]:
```
plt.hist(df2[X1], color = 'blue', edgecolor = 'black', bins = 4)

# Add labels
plt.title(f"{X1}, Вариант 4, 2020")
plt.xlabel('Уровень ВВП')
plt.ylabel('Количество стран')
```

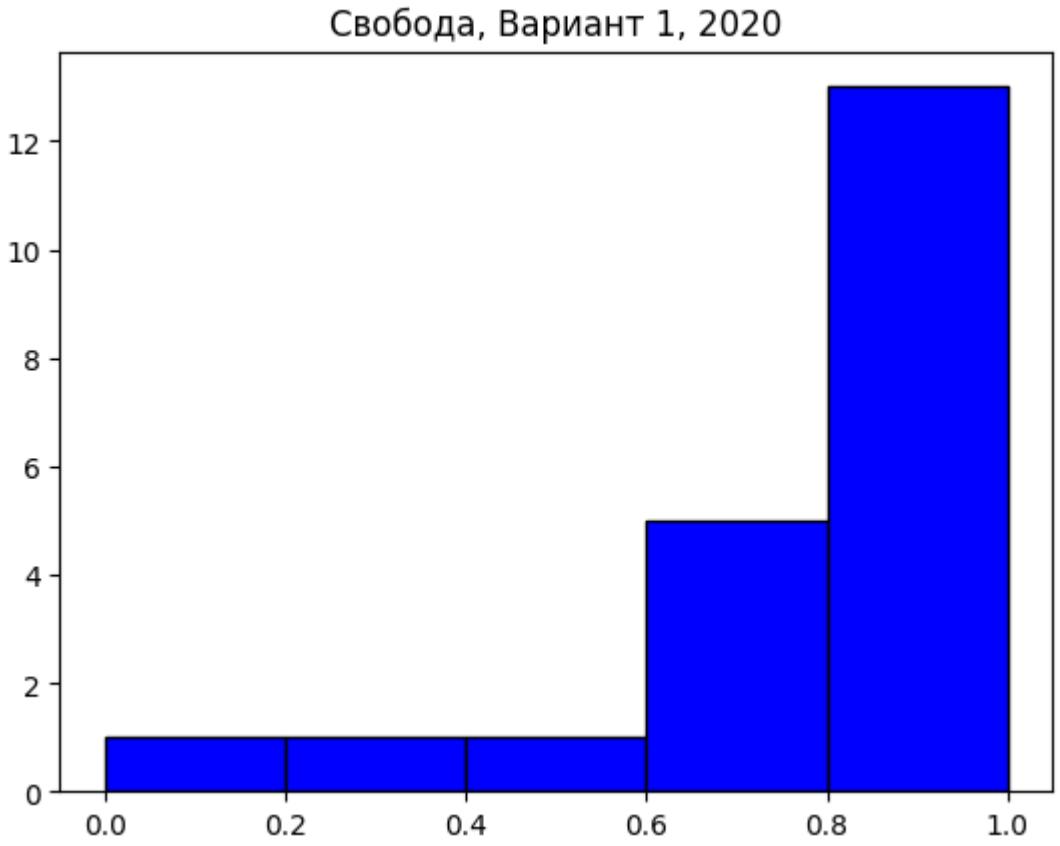
Out[92]:
```
Text(0, 0.5, 'Количество стран')
```



Распределение больше всего похоже на нормальное, но без правого "хвоста". Тест Шапиро-Уилка это так же не опровергает.

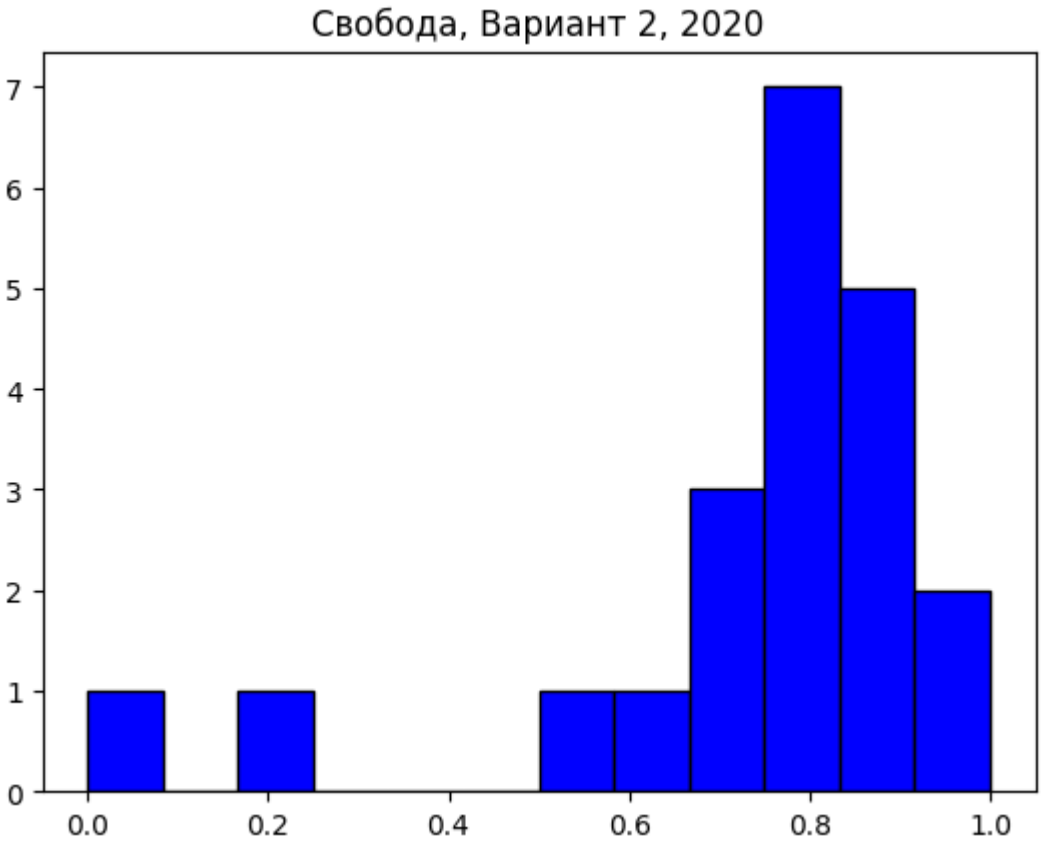
In [99]:
```
bins = 1 + int(np.log(len(df2[X2]))/np.log(2))
plt.hist(df2[X2], color = 'blue', edgecolor = 'black', bins = bins)

# Add labels
plt.title(f"{X2}, Вариант 1, 2020")
scipy.stats.shapiro(df2[X2])
```

Out[99]:
```
ShapiroResult(statistic=0.7289867997169495, pvalue=6.469947402365506e-05)
```



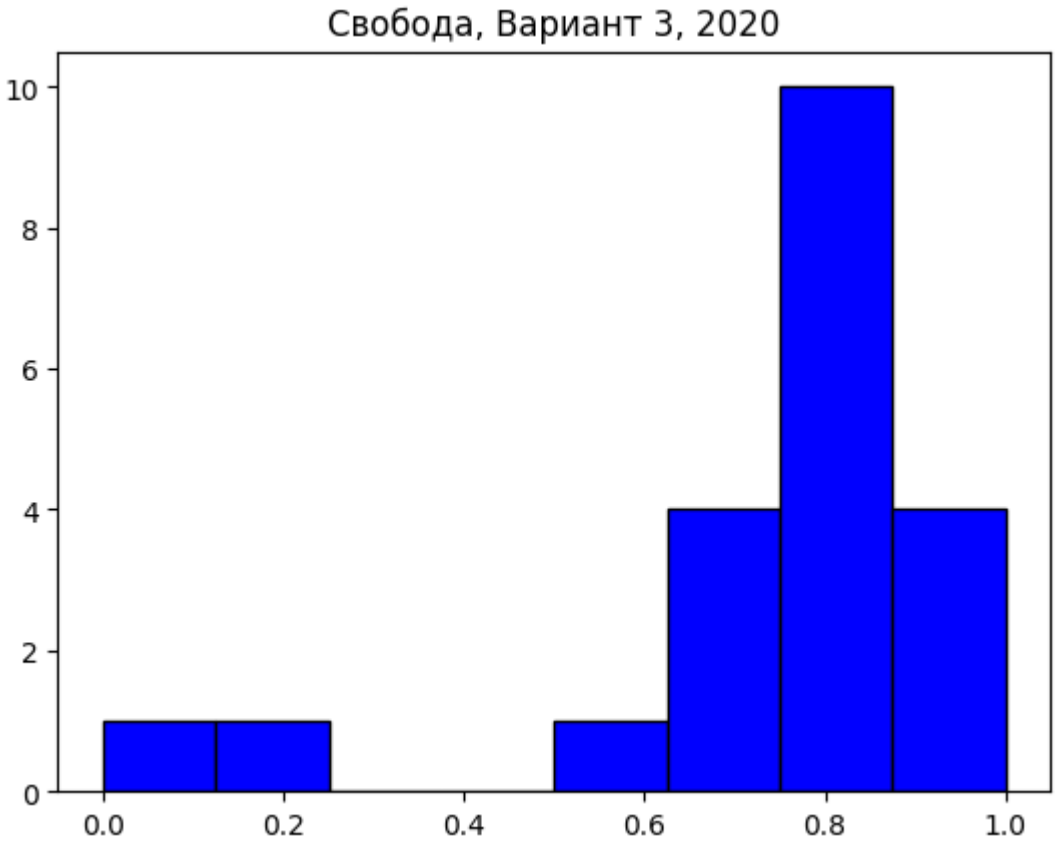
In [100…
```
plt.hist(df2[X2], color = 'blue', edgecolor = 'black', bins = 12)
plt.title(f"{X2}, Вариант 2, 2020")
```

Out[100…
```
Text(0.5, 1.0, 'Свобода, Вариант 2, 2020')
```



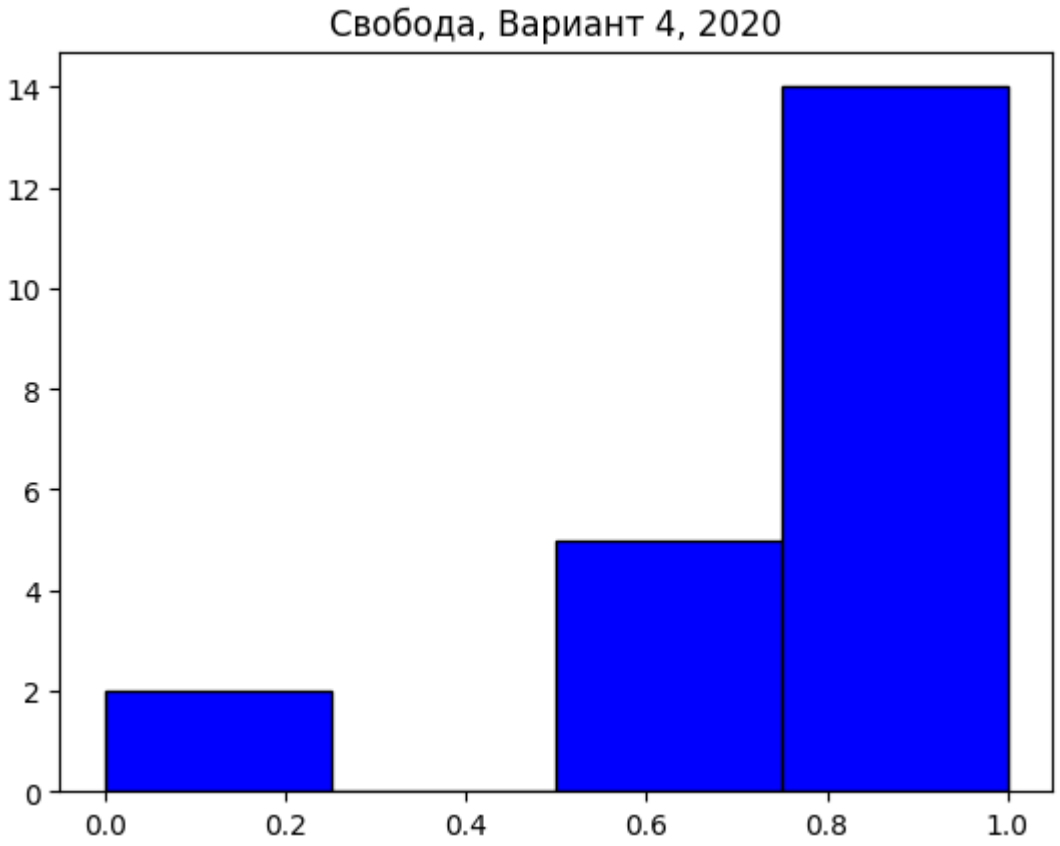
In [101…
```
plt.hist(df2[X2], color = 'blue', edgecolor = 'black', bins = 8)
plt.title(f"{X2}, Вариант 3, 2020")
```

Out[101…
```
Text(0.5, 1.0, 'Свобода, Вариант 3, 2020')
```



In [102…
```
plt.hist(df2[X2], color = 'blue', edgecolor = 'black', bins = 4)
plt.title(f"{X2}, Вариант 4, 2020")
```

Out[102…
```
Text(0.5, 1.0, 'Свобода, Вариант 4, 2020')
```



Распределение напоминает экспоненциальное. Лучше всего это выражено, когда количество карманов равно 5. Тест Шапиро-Уилка так же даёт нам право отклонить нулевую гипотезу о том, что распрделение нормальное.

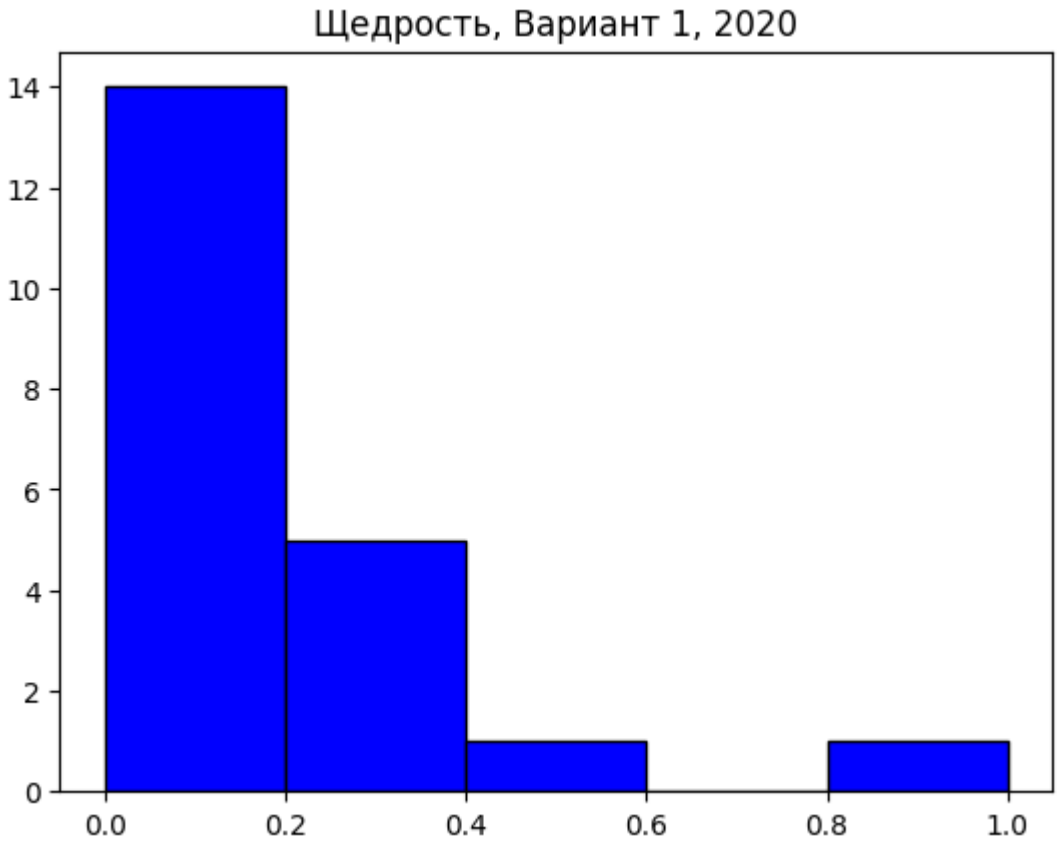
In [103…

```
bins = 1 + int(np.log(len(df2))/np.log(2))
print(bins)
plt.hist(df2[X3], color = 'blue', edgecolor = 'black', bins = bins)

plt.title(f"{X3}, Вариант 1, 2020")

scipy.stats.shapiro(df2[X1])
```

```
5
```

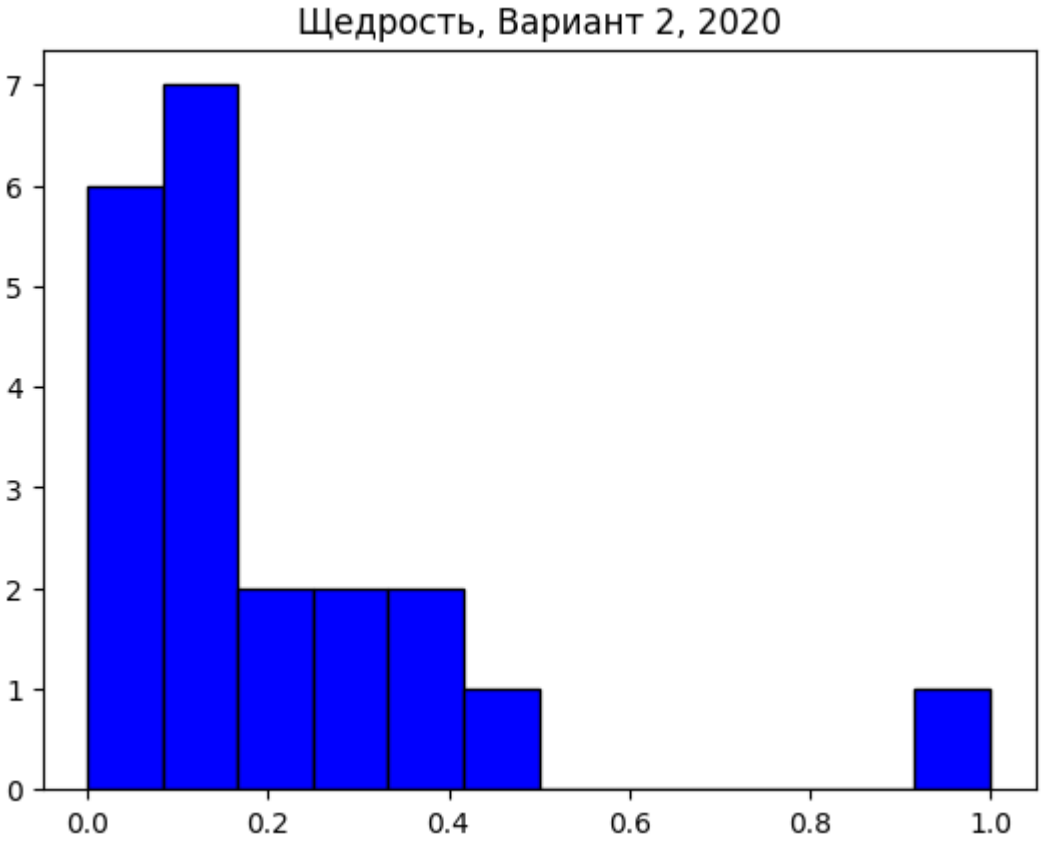
Out[103…

```
ShapiroResult(statistic=0.9281354546546936, pvalue=0.12629683315753937)
```



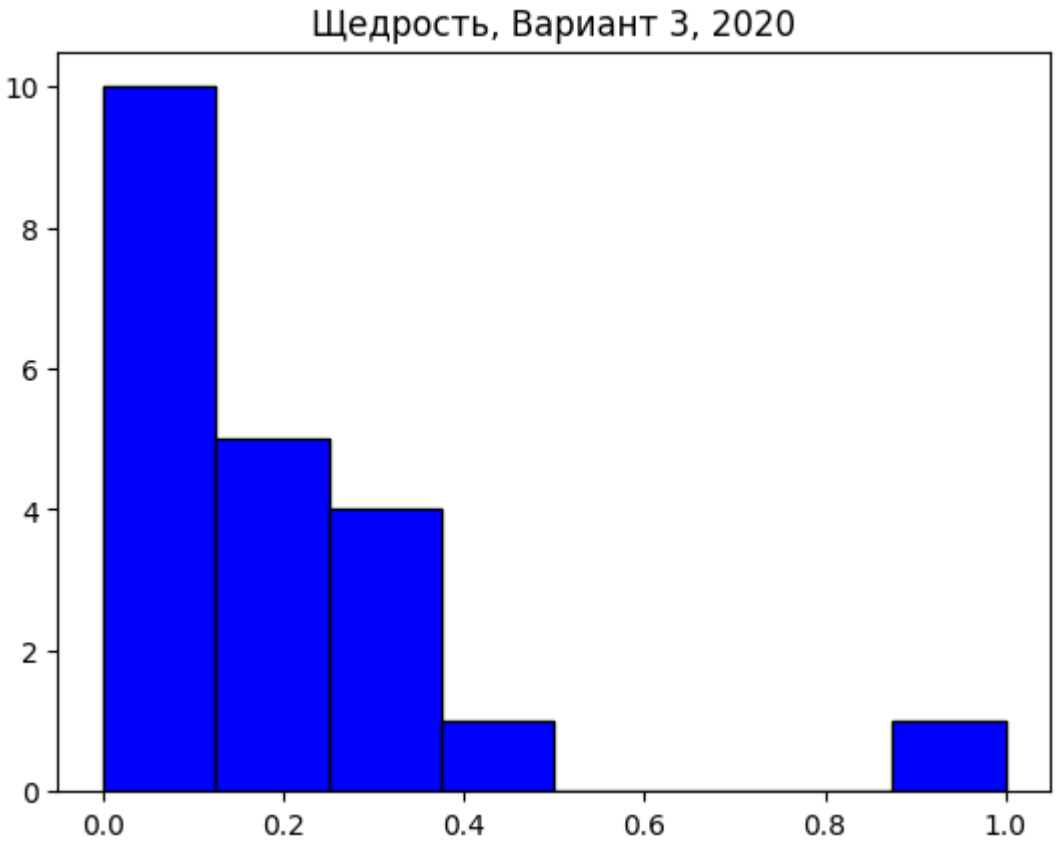
In [105…
```
plt.hist(df2[X3], color = 'blue', edgecolor = 'black', bins = 12)
plt.title(f"{X3}, Вариант 2, 2020")
```

Out[105…
```
Text(0.5, 1.0, 'Щедрость, Вариант 2, 2020')
```



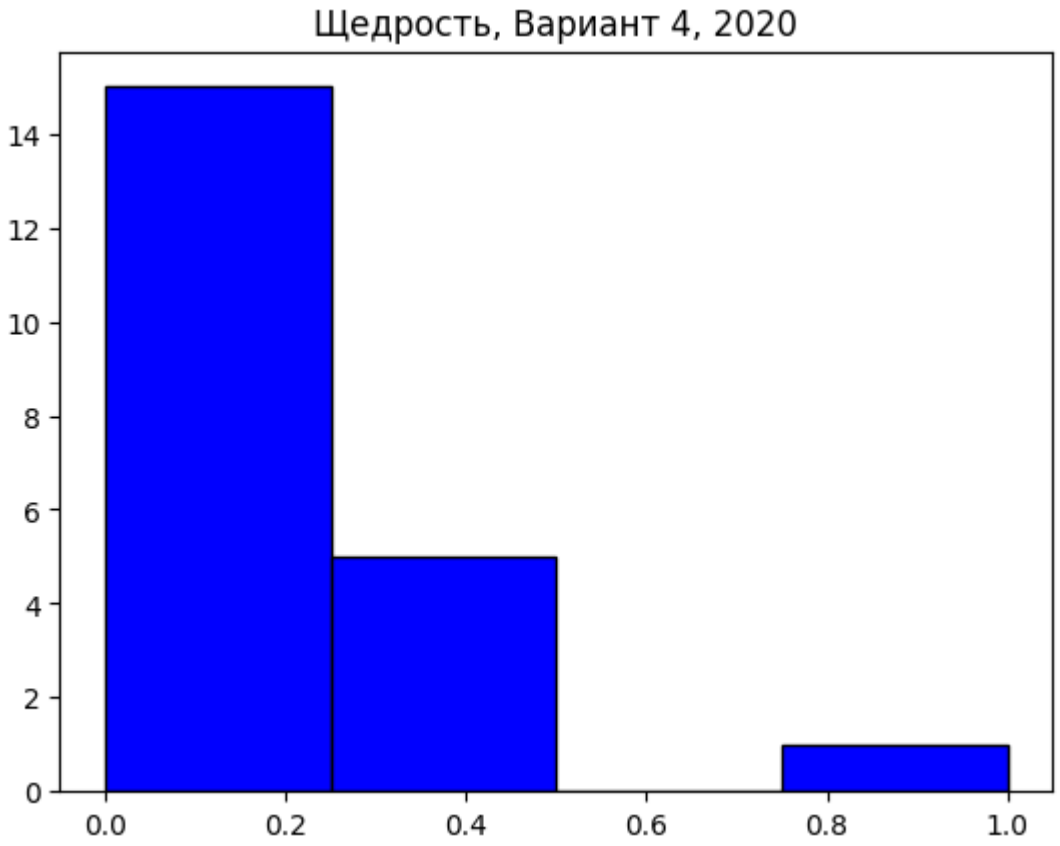
In [106…
```
plt.hist(df2[X3], color = 'blue', edgecolor = 'black', bins = 8)
plt.title(f"{X3}, Вариант 3, 2020")
```

Out[106…
```
Text(0.5, 1.0, 'Щедрость, Вариант 3, 2020')
```



In [107…
```
plt.hist(df2[X3], color = 'blue', edgecolor = 'black', bins = 4)
plt.title(f"{X3}, Вариант 4, 2020")
```

Out[107…
```
Text(0.5, 1.0, 'Щедрость, Вариант 4, 2020')
```

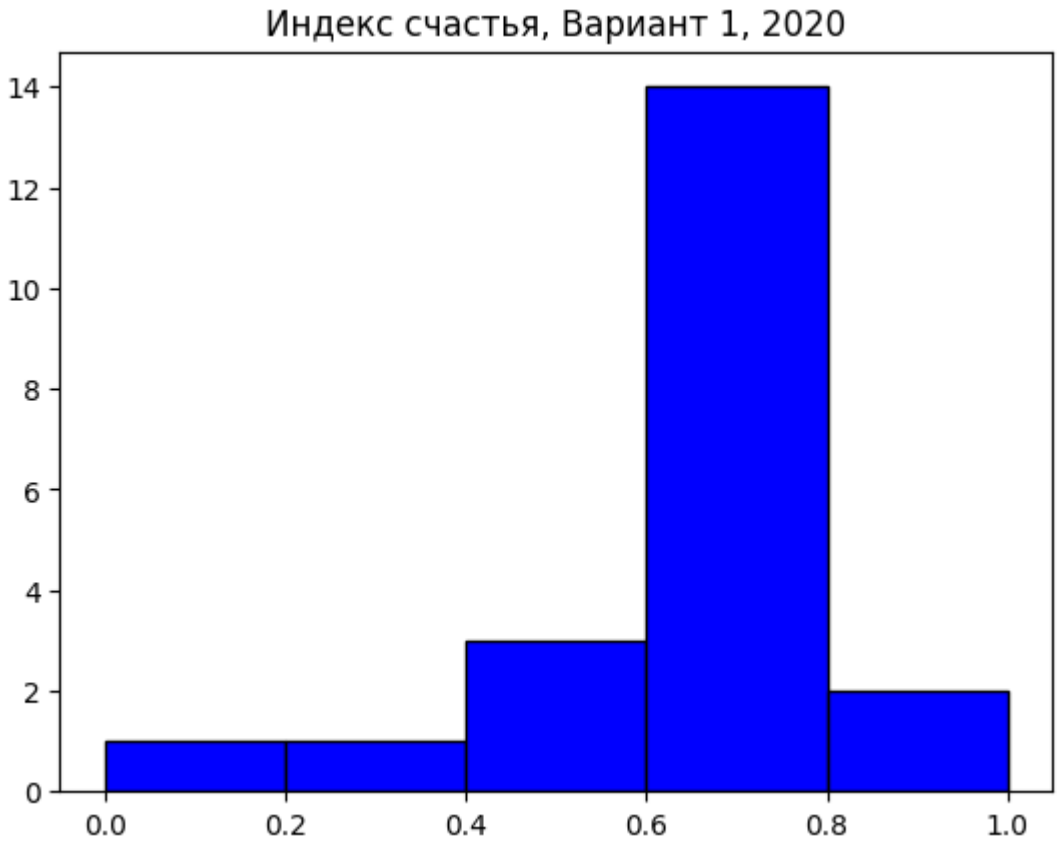


Распределение напоминает экспоненциальное. Лучше всего это показано на третьем варианте.

In [114…
```
bins = 1 + int(np.log(len(df2))/np.log(2))
print(bins)
plt.hist(df2["Индекс счастья"], color = 'blue', edgecolor = 'black', bins = bins)

plt.title("Индекс счастья, Вариант 1, 2020")

scipy.stats.shapiro(df2['Индекс счастья'])
```

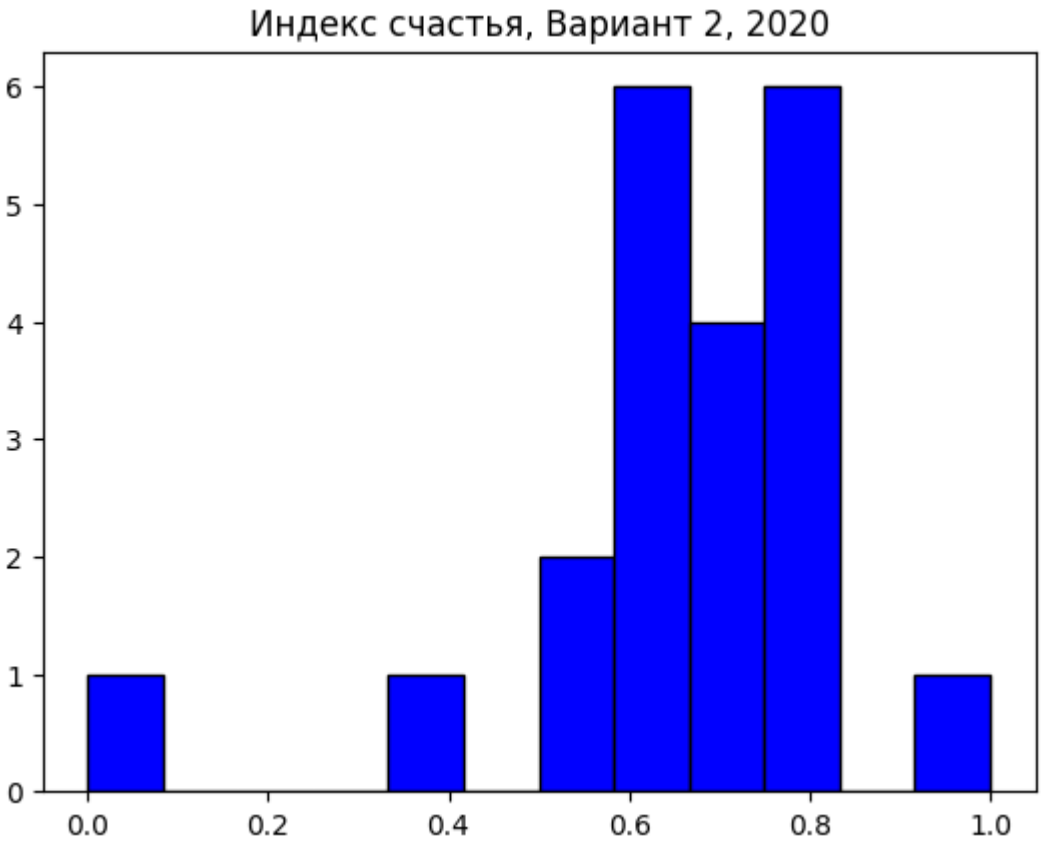
```
5
```

Out[114…
```
ShapiroResult(statistic=0.8053651452064514, pvalue=0.0007933441665954888)
```



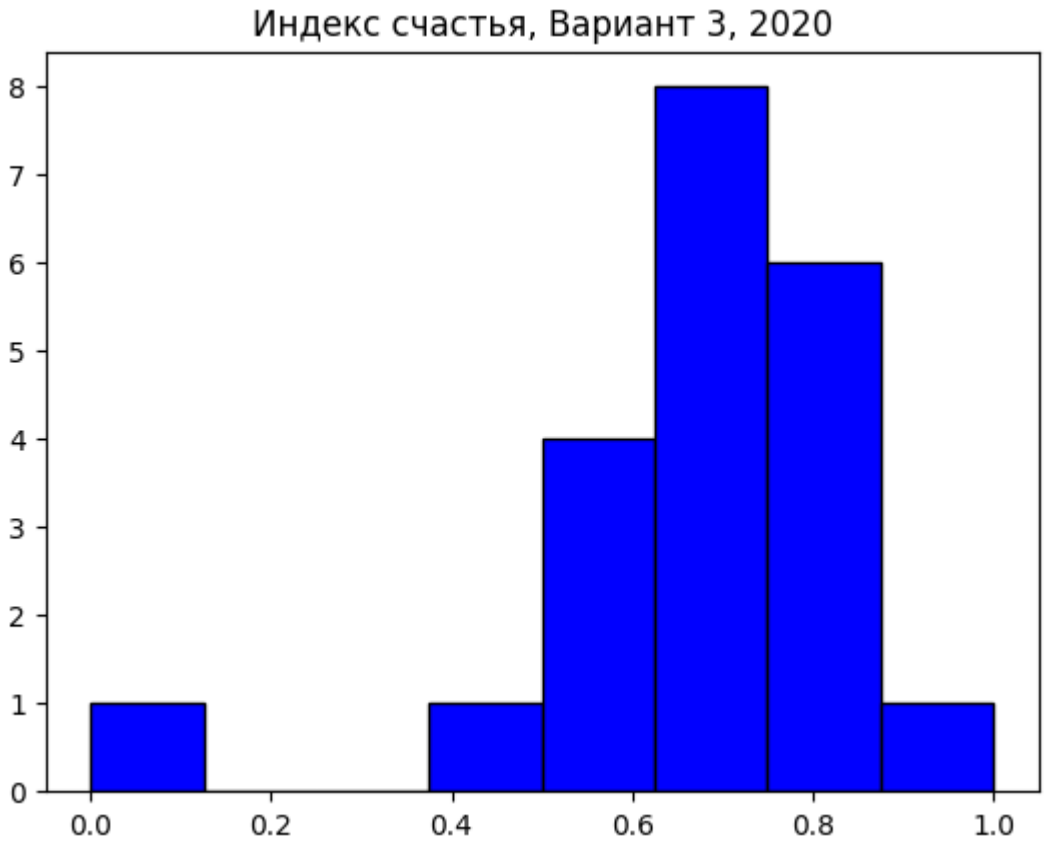
In [115…
```
plt.hist(df2["Индекс счастья"], color = 'blue', edgecolor = 'black', bins = 12)

plt.title("Индекс счастья, Вариант 2, 2020")
```

Out[115…
```
Text(0.5, 1.0, 'Индекс счастья, Вариант 2, 2020')
```



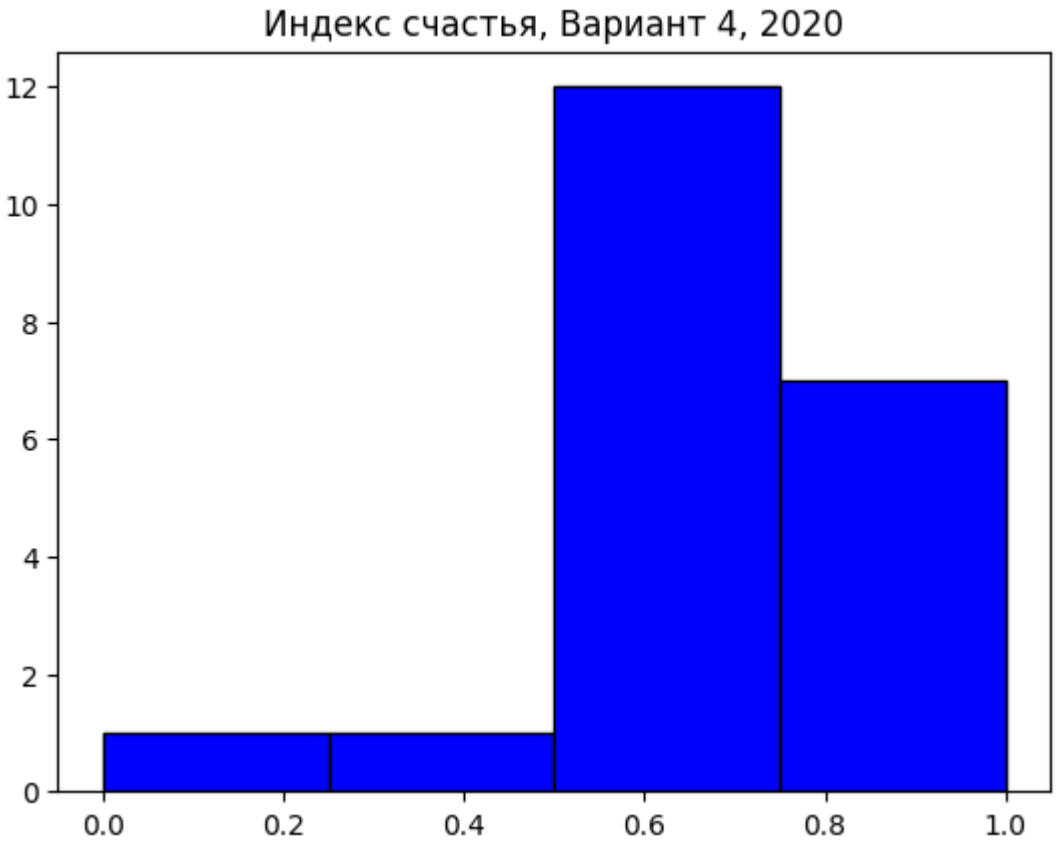
In [116…
```
plt.hist(df2["Индекс счастья"], color = 'blue', edgecolor = 'black', bins = 8)

plt.title("Индекс счастья, Вариант 3, 2020")
```

Out[116…
```
Text(0.5, 1.0, 'Индекс счастья, Вариант 3, 2020')
```



In [117…
```
plt.hist(df2["Индекс счастья"], color = 'blue', edgecolor = 'black', bins = 4)

plt.title("Индекс счастья, Вариант 4, 2020")
```

Out[117…
```
Text(0.5, 1.0, 'Индекс счастья, Вариант 4, 2020')
```



Распределение на 2 и 4 изображении не похожи на известные нам, однако на 1 и 3 отдаленно напоминают нормальное. Тест Шапиро-Уилка даёт нам право отклонить нулевую гипотезу о нормальном распределнии случайной величины, однако варианты 1 и 3 его отдаленно напоминают. Можем предположить, что распределение все-таки нормальное.

Задание №2.

In [195…

```
def describe_data(df):
    mean = np.round(np.mean(df),4)
    if len(set(df))==len(df):
        moda='-'
    else:
        moda = np.round(stat.mode(df),4)
    median = np.round(np.median(df), 4)
    var = np.round(np.var(df), 4)
    s = np.round(var**0.5,4)
    skew = np.round(stat.skew(df), 4)
    kurt = np.round(stat.kurtosis(df),4)
    return [mean, moda, median, var, s, skew, kurt]
```

In [196…

```
indexes = ["Выборочное среднее", "Выборочная мода",
           "Выборочная медиана", "Несмещённая выборочная дисперсия",
```

```
            "Несмещённое квадратическое отклонение",
            "Выборочный коэффициент симметрии",
            "Выборочный коэффициент эксцесса"]

columns = [f"{X1}, 2016", f"{X1}, 2020",
           f"{X2}, 2016", f"{X2}, 2020",
            f"{X3}, 2016", f"{X3}, 2020",
            f"{Y}, 2016", f"{Y}, 2020"]

df_new = pd.DataFrame(index=indexes, columns=columns)
for i in range(4):
    a1 = [float(i) for i in df.iloc[:,[i]].values]
    a2 = [float(i) for i in df2.iloc[:,[i]].values]
    v1 = pd.DataFrame(describe_data(a1))
    v2 = pd.DataFrame(describe_data(a2))
    df_new.iloc[:,[2*i]] = v1
    df_new.iloc[:,[2*i+1]] = v2
df_new
```

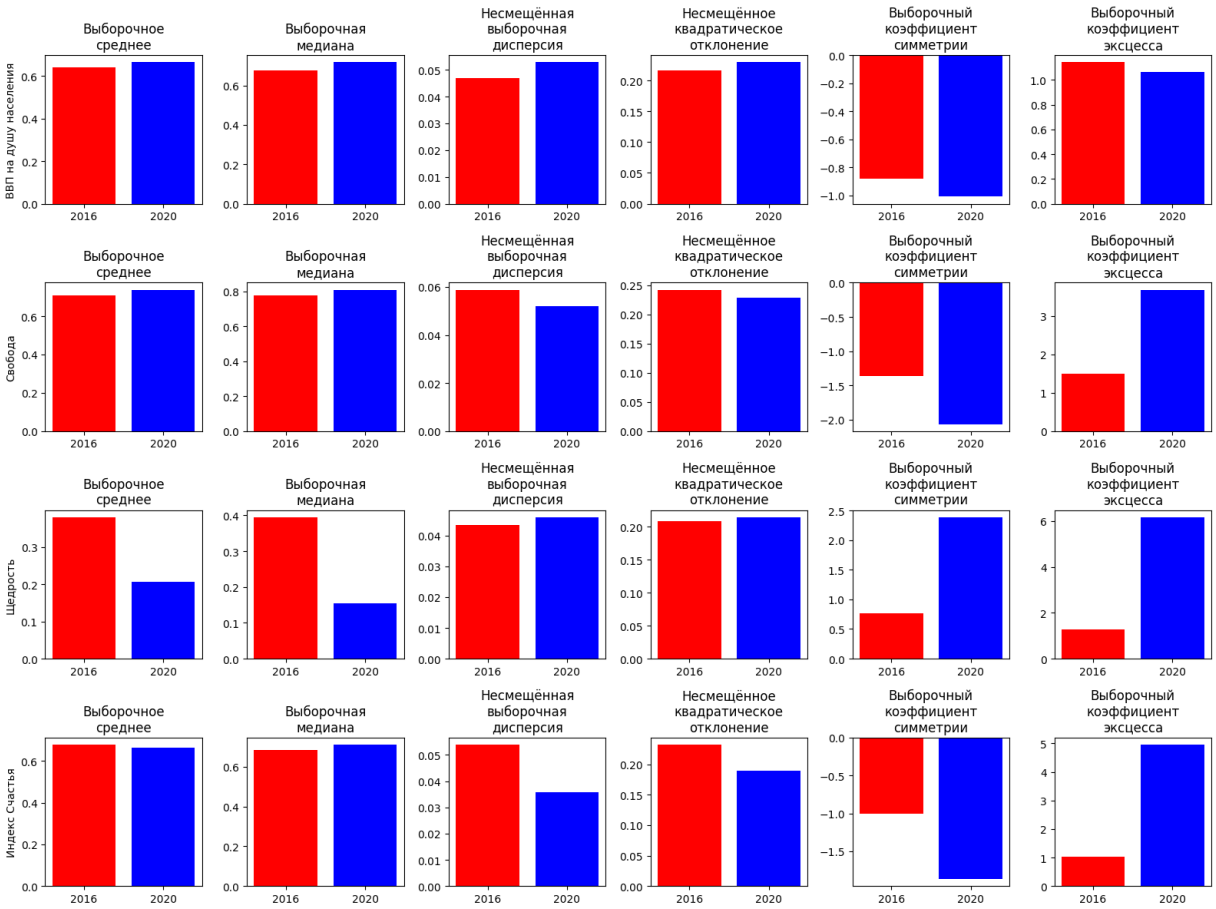
Out[196…

	ВВП на душу населения, 2016	ВВП на душу населения, 2020	Свобода, 2016	Свобода, 2020	Щедрость, 2016	Щедрость, 2020	С
Выборочное среднее	0.6406	0.6641	0.7089	0.7378	0.3796	0.2066	
Выборочная мода	-	-	-	-	-	-	
Выборочная медиана	0.6757	0.7179	0.7743	0.8084	0.3945	0.1553	
Несмещённая выборочная дисперсия	0.047	0.0528	0.0588	0.0521	0.0435	0.046	
Несмещённое квадратическое отклонение	0.2168	0.2298	0.2425	0.2283	0.2086	0.2145	
Выборочный коэффициент симметрии	-0.8787	-1.0091	-1.3618	-2.0641	0.7721	2.3891	
Выборочный коэффициент эксцесса	1.142	1.0633	1.4865	3.6773	1.2719	6.1678	

In [197…

```
columns=[X1,X2,X3,Y]
indeces = [0,2,3,4,5,6]
fi, axs = plt.subplots(len(columns), len(indeces),  figsize= (16,12))
for i in range(len(columns)):
    for j in range(len(indeces)):
        axs[i, 0].set_ylabel(columns[i])
        val = df_new.iloc[indeces[j], [2*i, 2*i+1]].values.flatten()
```

```
        axs[i,j].bar(range(2), val, color=['red', 'blue'])
        title = df_new.index.values.flatten()[int(indeces[j])].replace(" ", "\n")
        axs[i,j].set_title(title)
        axs[i,j].set_xticks(list(range(2)))
        axs[i,j].set_xticklabels(['2016','2020'])
plt.tight_layout()
plt.show()
```



Вывод:

1. ВВП на душу населения: практически не изменился, но все же незначительно вырос. Медиана и десперсия так же немного увеличились.
2. Свобода: характеристики практически не изменились
3. Щедрость: средие характеристики уменьшились, а дисперсия не изменилась |=> щедрость сместилась влево
4. Индекс счастья: дисперсия уменьшилась |=> стала более стабильной

Задание №3

In [223…

```
df_new["ВВП на душу населения, 2016"]
```

Out[223…

```
Выборочное среднее                          0.6406
Выборочная мода                                  -
Выборочная медиана                          0.6757
Несмещённая выборочная дисперсия             0.047
Несмещённое квадратическое отклонение       0.2168
Выборочный коэффициент симметрии           -0.8787
Выборочный коэффициент эксцесса             1.142
Name: ВВП на душу населения, 2016, dtype: object
```

In [226…

```
#df_new.iloc[[],[]]
q95 = stat.norm.ppf((0.95+1)/2)
q99 = stat.norm.ppf((0.99+1)/2)
np.round([q95,q99],4)
#array([1.96 , 2.5758])
n = 20
indexes = ["Выборочное среднее",
           "Несмещённое квадратическое отклонение",
           "Точность оценки при уровне значимости 0.95",
           "Нижняя граница доверительного интервала для оценки математического ожид
           "Верхняя граница доверительного интервала для оценки математического ожи
           "Точность оценки при уровне значимости 0.99",
           "Нижняя граница доверительного интервала для оценки математического ожид
           "Верхняя граница доверительного интервала для оценки математического ожи
task3 = pd.DataFrame(columns=df_new.iloc[:,[0]].columns.values.flatten(),
index=indexes)
sigma = float(df_new.iloc[[4],0].values.flatten())
mu = float(df_new.iloc[[0],0].values.flatten())
eps95 = q95*sigma/(n**0.5)
eps99 = q99*sigma/(n**0.5)
arr = pd.DataFrame([mu,sigma, eps95, mu-eps95, mu+eps95, eps99, mu-eps99, mu+eps99]
task3.iloc[:,[0]]=arr
np.round(task3.astype(float), 4)
```

Out[226…

	ВВП на душу населения, 2016
Выборочное среднее	0.6406
Несмещённое квадратическое отклонение	0.2168
Точность оценки при уровне значимости 0.95	0.0950
Нижняя граница доверительного интервала для оценки математического ожидания при уровне значимости 0.95	0.5456
Верхняя граница доверительного интервала для оценки математического ожидания при уровне значимости 0.95	0.7356
Точность оценки при уровне значимости 0.99	0.1249
Нижняя граница доверительного интервала для оценки математического ожидания при уровне значимости 0.99	0.5157
Верхняя граница доверительного интервала для оценки математического ожидания при уровне значимости 0.99	0.7655

Вывод: чем больше уровень значимости, тем больше доверительный интервал, а значит и точность оценки. Но в связи с увеличением доверительного интервала полезность этой оценки снижается

Задание №4

In [210…

```
print('2016:')
display(df.corr(method = "pearson"))
print('2020:')
display(df2.corr(method = "pearson"))

def local_analys(x):
    if abs(x)>=0.1 and abs(x)<0.3:
        return "Слабая"
    elif abs(x)>=0.3 and abs(x)<0.5:
        return "Умеренная"
    elif abs(x)>=0.5 and abs(x)<0.7:
        return "Заметная"
    elif abs(x)>=0.7 and abs(x)<0.9:
        return "Высокая"
    elif abs(x)>=0.9 and abs(x)<=0.99:
        return "Весьма высокая"
    elif abs(x)==1:
        return '-'
    elif abs(x)<0.1:
        return "Практически отсутсвует"
    else:
        return None
print("2016:")
display(df.corr(method='pearson').applymap(local_analys))
print("2020:")
display(df2.corr(method='pearson').applymap(local_analys))
```

2016:

	ВВП на душу населения	Свобода	Щедрость	Индекс Счастья
ВВП на душу населения	1.000000	0.431034	-0.418176	0.777641
Свобода	0.431034	1.000000	-0.059208	0.512339
Щедрость	-0.418176	-0.059208	1.000000	-0.402830
Индекс Счастья	0.777641	0.512339	-0.402830	1.000000

2020:

	ВВП на душу населения	Свобода	Щедрость	Индекс счастья
ВВП на душу населения	1.000000	0.480838	-0.696882	0.659859
Свобода	0.480838	1.000000	-0.485188	0.817212
Щедрость	-0.696882	-0.485188	1.000000	-0.628030
Индекс счастья	0.659859	0.817212	-0.628030	1.000000

```
2016:
```

	ВВП на душу населения	Свобода	Щедрость	Индекс Счастья
ВВП на душу населения	-	Умеренная	Умеренная	Высокая
Свобода	Умеренная	-	Практически отсутсвует	Заметная
Щедрость	Умеренная	Практически отсутсвует	-	Умеренная
Индекс Счастья	Высокая	Заметная	Умеренная	-

```
2020:
```

	ВВП на душу населения	Свобода	Щедрость	Индекс счастья
ВВП на душу населения	-	Умеренная	Заметная	Заметная
Свобода	Умеренная	-	Умеренная	Высокая
Щедрость	Заметная	Умеренная	-	Заметная
Индекс счастья	Заметная	Высокая	Заметная	-

Задание №5

In [73]:
```
print('2016:')
display(df.pcorr())
print('2020:')
display(df2.pcorr())
```

```
2016:
```

	ВВП на душу населения	Свобода	Щедрость	Индекс Счастья
ВВП на душу населения	1.000000	0.097895	-0.197487	0.652774
Свобода	0.097895	1.000000	0.202006	0.339172
Щедрость	-0.197487	0.202006	1.000000	-0.193766
Индекс Счастья	0.652774	0.339172	-0.193766	1.000000

2020:

	ВВП на душу населения	Свобода	Щедрость	Индекс счастья
ВВП на душу населения	1.000000	-0.119773	-0.479967	0.353937
Свобода	-0.119773	1.000000	-0.003023	0.741738
Щедрость	-0.479967	-0.003023	1.000000	-0.207079
Индекс счастья	0.353937	0.741738	-0.207079	1.000000

Вывод:

1. ВВП на душу населения сильно коррелирует с индексом счастья в 2016 году и в почти в 2 раза меньше в 2020. Корреляция с остальными переменными меньше, только в 2020 году он коррелирует с щедростью.

2. Свобода немного коррелирует с индексом счастья в 2016 году и сильно коррелирует в 2020.

3. Щедрость слабо коррелирует с чем-либо в 2016 году, при этом наблюдается заметный прирост корреляции в 2020.

4. Индекс счастья значитлеьно коррелирует с ВВП на душу населения в 2016 году и есть небольшая корреляция со свободой. В 2020 году корреляция с ВВП упала почти в 2 раза, и почти в 2 раза выросла со свободой.

Задание №6

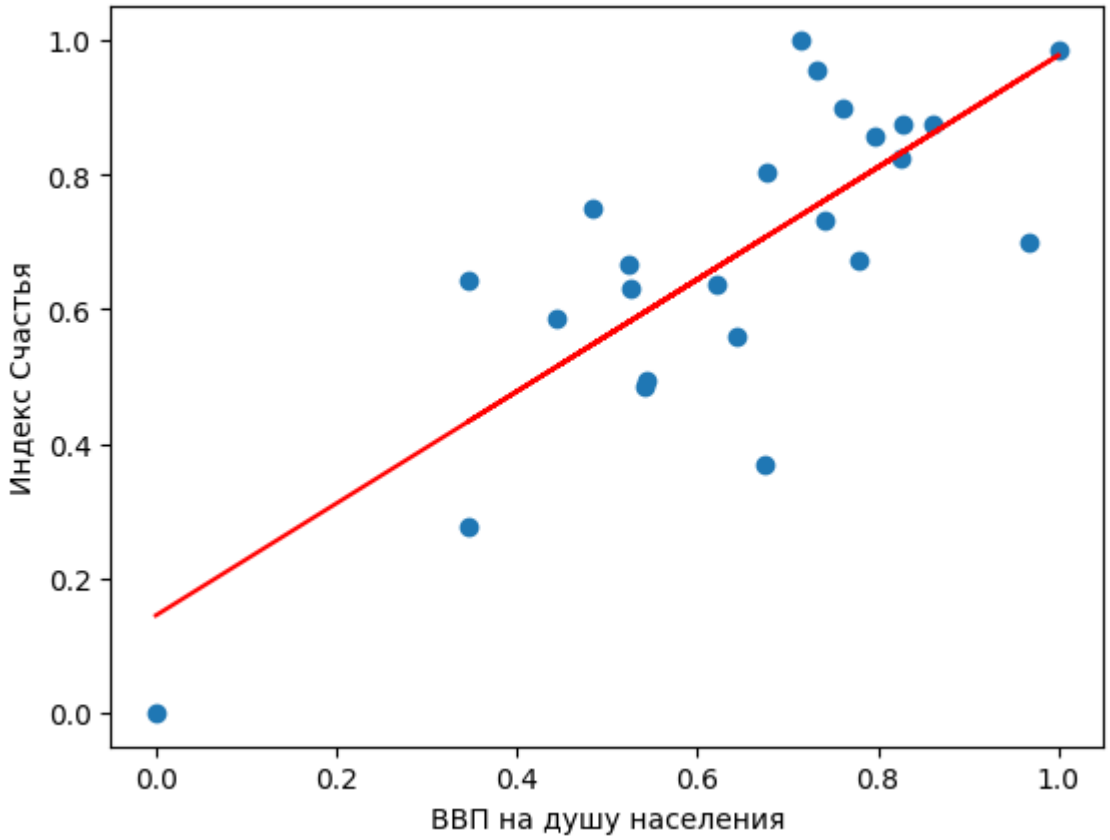
In [94]:
```
modelx1 = LinearRegression()
modelx2 = LinearRegression()
modelx3 = LinearRegression()

x1_data = df[X1].to_numpy().reshape(-1,1)
x2_data = df[X2].to_numpy().reshape(-1,1)
x3_data = df[X3].to_numpy().reshape(-1,1)
y_data = df[Y].to_numpy()

modelx1.fit(x1_data, y_data)
modelx2.fit(x2_data, y_data)
modelx3.fit(x3_data, y_data)
```

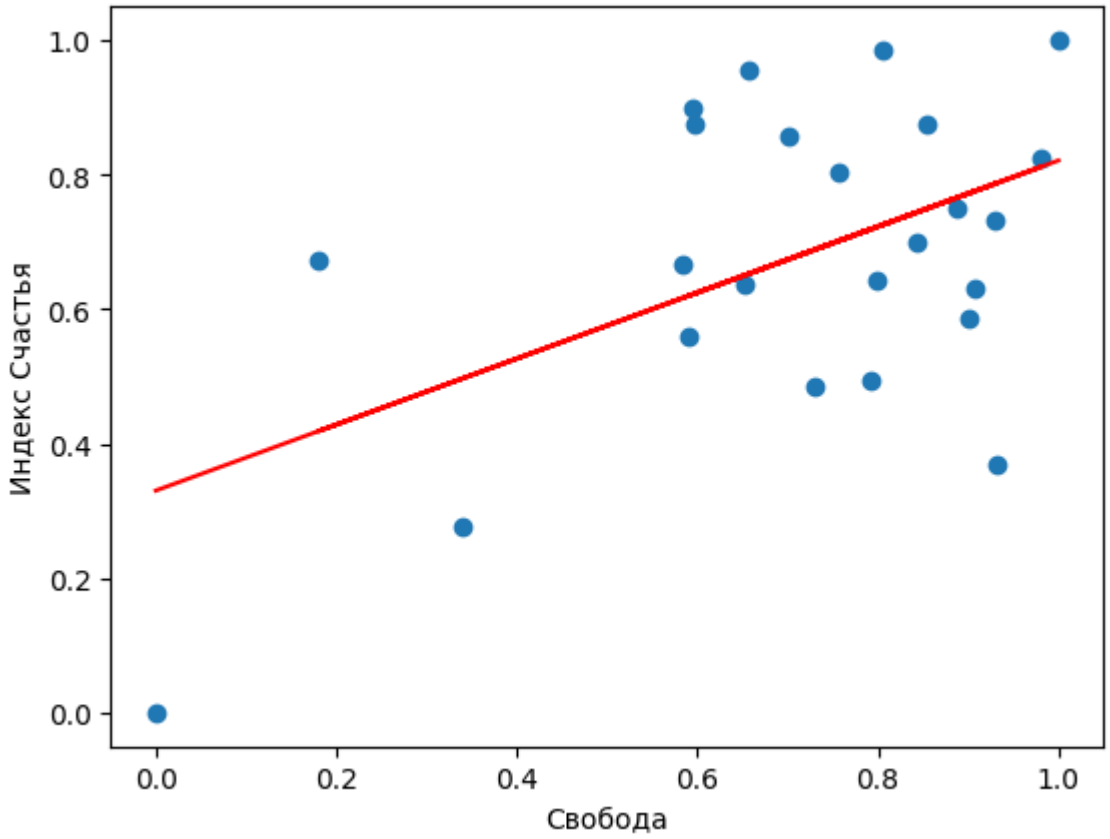
```
print("--------x1--------")
print('coefficient of determination(R_squared):', modelx1.score(x1_data, y_data))
print('intercept:', modelx1.intercept_)
print('slope:', modelx1.coef_[0])
print("--------x2--------")
print('coefficient of determination(R_squared):', modelx2.score(x2_data, y_data))
print('intercept:', modelx2.intercept_)
print('slope:', modelx2.coef_[0])
print("--------x3--------")
print('coefficient of determination(R_squared):', modelx3.score(x3_data, y_data))
print('intercept:', modelx3.intercept_)
print('slope:', modelx3.coef_[0])
```

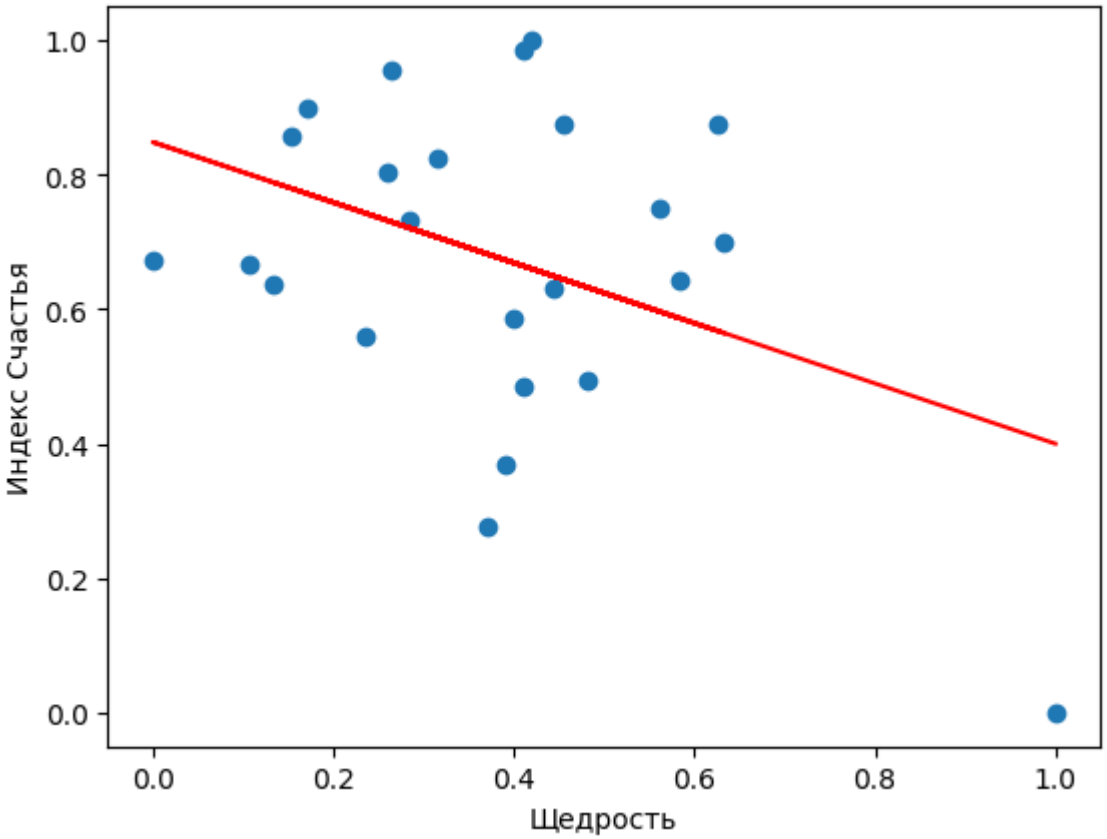
```
--------x1--------
coefficient of determination(R_squared): 0.6047260392095599
intercept: 0.14452121115261873
slope: 0.8326327427725743
--------x2--------
coefficient of determination(R_squared): 0.2624909921304829
intercept: 0.33009987816462966
slope: 0.4906208283645696
--------x3--------
coefficient of determination(R_squared): 0.16227170233541965
intercept: 0.8481105345460422
slope: -0.4483135431749033
```

In [95]:

```
# ==================1
fig, ax = plt.subplots(1)

ax.scatter(x = df[X1], y = df[Y])
plt.plot(df[X1], modelx1.predict(x1_data), color="red")
ax.set_xlabel(X1)
ax.set_ylabel(Y)

plt.show()
# ==================2
fig, ax = plt.subplots(1)

ax.scatter(x = df[X2], y = df[Y])
plt.plot(df[X2], modelx2.predict(x2_data), color="red")
ax.set_xlabel(X2)
ax.set_ylabel(Y)

plt.show()
# ==================3
fig, ax = plt.subplots(1)

ax.scatter(x = df[X3], y = df[Y])
plt.plot(df[X3], modelx3.predict(x3_data), color="red")
ax.set_xlabel(X3)
ax.set_ylabel(Y)

plt.show()
```


Индекс Счастья
ВВП на душу населения
0.0
0.2
0.4
0.6
0.8
1.0


Индекс Счастья
Свобода
0.0
0.2
0.4
0.6
0.8
1.0



In [98]:
```
import statsmodels.api as sm

x1_data = sm.add_constant(x1_data)
modelx1 = sm.OLS(y_data, x1_data).fit()

x2_data = sm.add_constant(x2_data)
modelx2 = sm.OLS(y_data, x2_data).fit()

x3_data = sm.add_constant(x3_data)
modelx3 = sm.OLS(y_data, x3_data).fit()

print("----------------------------------------x1----------------------------------")
print(modelx1.summary())
print("----------------------------------------x2----------------------------------")
print(modelx2.summary())
print("----------------------------------------x3----------------------------------")
print(modelx3.summary())
```

```
---------------------------------------x1---------------------------------------
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.605
Model:                            OLS   Adj. R-squared:                  0.587
Method:                 Least Squares   F-statistic:                     33.66
Date:                Sat, 23 Dec 2023   Prob (F-statistic):           7.76e-06
Time:                        18:54:51   Log-Likelihood:                 12.124
No. Observations:                  24   AIC:                            -20.25
Df Residuals:                      22   BIC:                            -17.89
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1445      0.097      1.489      0.151      -0.057       0.346
x1             0.8326      0.144      5.802      0.000       0.535       1.130
==============================================================================
Omnibus:                        0.406   Durbin-Watson:                   0.985
Prob(Omnibus):                  0.816   Jarque-Bera (JB):                0.408
Skew:                          -0.266   Prob(JB):                        0.816
Kurtosis:                       2.647   Cond. No.                         6.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
---------------------------------------x2---------------------------------------
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.262
Model:                            OLS   Adj. R-squared:                  0.229
Method:                 Least Squares   F-statistic:                     7.830
Date:                Sat, 23 Dec 2023   Prob (F-statistic):             0.0105
Time:                        18:54:51   Log-Likelihood:                 4.6395
No. Observations:                  24   AIC:                            -5.279
Df Residuals:                      22   BIC:                            -2.923
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3301      0.131      2.513      0.020       0.058       0.603
x1             0.4906      0.175      2.798      0.010       0.127       0.854
==============================================================================
Omnibus:                        1.172   Durbin-Watson:                   0.412
Prob(Omnibus):                  0.557   Jarque-Bera (JB):                0.915
Skew:                          -0.191   Prob(JB):                        0.633
Kurtosis:                       2.123   Cond. No.                         6.28
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
---------------------------------------x3---------------------------------------
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.162
Model:                            OLS   Adj. R-squared:                  0.124
Method:                 Least Squares   F-statistic:                     4.261
Date:                Sat, 23 Dec 2023   Prob (F-statistic):             0.0510
Time:                        18:54:51   Log-Likelihood:                 3.1106
No. Observations:                  24   AIC:                            -2.221
Df Residuals:                      22   BIC:                            0.1350
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8481      0.094      9.015      0.000       0.653       1.043
x1            -0.4483      0.217     -2.064      0.051      -0.899       0.002
==============================================================================
Omnibus:                        1.095   Durbin-Watson:                   0.337
Prob(Omnibus):                  0.578   Jarque-Bera (JB):                0.907
Skew:                          -0.217   Prob(JB):                        0.635
Kurtosis:                       2.153   Cond. No.                         5.51
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Вывод по моделям линейной регрессии:

1. ВВП на душу населения. Модель предсказывает примерно 60% данных, что можно назвать не совсем плохим результатом. P-value для F-теста меньше 0.05, из-за чего мы можем говорить о статической значимости модели. Коэффициенты B0 и B1 так же являются статически значимыми.
2. Свобода. Модель предсказывает примерно 26% данных, что можно называть плохим результатом. P-value для F-теста меньше 0.05, из-за чего мы можем говорить о статической значимости модели. Коэффициенты B0 и B1 так же являются статически значимыми.
3. Щедрость. Модель предсказывает примерно 16% данных, что можно называть плохим результатом. P-value для F-теста меньше 0.05, из-за чего мы можем говорить о статической значимости модели. Коэффицент B0 является статически значимыми, p-значение для B1 0.051, значит, его можно назвать не статистически значимым.

Задание №7

2016:

In [42]:
```
from sklearn.model_selection import train_test_split

x123_data = df[[X1, X2, X3]]
y_data = df[Y]
```

```
X_train, X_test, y_train, y_test = train_test_split(x123_data, y_data, test_size=0.
```

In [43]:
```
modelx123 = LinearRegression()

modelx123.fit(X_train, y_train)

coeff_df = pd.DataFrame(modelx123.coef_, x123_data.columns, columns=['Coefficient']
coeff_df
```

Out[43]:

	Coefficient
ВВП на душу населения	0.807415
Свобода	0.204839
Щедрость	-0.016073

In [44]:
```
print('coefficient of determination(R_squared):', modelx123.score(X_train, y_train)
print('intercept:', modelx123.intercept_)
print('slope:', modelx123.coef_[0])
```

```
coefficient of determination(R_squared): 0.6970768382329859
intercept: 0.03309005339161131
slope: 0.8074154818765065
```

In [45]:
```
y_pred = modelx123.predict(X_test)
```

In [46]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred
```

```
Mean Absolute Error: 0.27671891546702276
Mean Squared Error: 0.0765733581772453
Root Mean Squared Error: 0.27671891546702276
```

In [70]:
```
x1_data = df[X1].to_numpy().reshape(-1, 1)
x2_data = df[X2].to_numpy().reshape(-1, 1)
x3_data = df[X3].to_numpy().reshape(-1, 1)
y_data = df["Индекс Счастья"].to_numpy()

X1_ = sm.add_constant(x1_data)
X2_ = sm.add_constant(x2_data)
X3_ = sm.add_constant(x3_data)

X = sm.add_constant(df[[X1, X2, X3]])


est = sm.OLS(y_data, X)
est2 = est.fit()

print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.657
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     12.75
Date:                Thu, 28 Dec 2023   Prob (F-statistic):           6.98e-05
Time:                        02:26:38   Log-Likelihood:                 13.814
No. Observations:                  24   AIC:                            -19.63
Df Residuals:                      20   BIC:                            -14.92
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1452      0.145      0.998      0.330      -0.158       0.449
ВВП на душу населения     0.6657      0.173      3.854      0.001       0.305       1.026
Свобода        0.2267      0.141      1.612      0.123      -0.067       0.520
Щедрость      -0.1434      0.162     -0.883      0.388      -0.482       0.195
==============================================================================
Omnibus:                        3.418   Durbin-Watson:                   0.937
Prob(Omnibus):                  0.181   Jarque-Bera (JB):                1.841
Skew:                          -0.614   Prob(JB):                        0.398
Kurtosis:                       3.575   Cond. No.                         11.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Вывод:

1. ВВП на душу населения. Коэффицент этого параемтра равен 0.66, коэффицент стат.значим, по-скольку p-value равно 0.001.
2. Свобода. Коэффицент этого параемтра равен 0.23, коэффицент не стат.значим, по-скольку p-value равно 0.123.
3. Щедрость. Коэффицент этого параемтра равен -0.14, коэффицент не стат.значим, по-скольку p-value равно 0.388, попробуем убрать его и посмотрим на результат.
4. Константа B0. Её p-value равно 0.33, но убирать ее из модели будем в последнюю очередь.
5. Модель предсказывает примерно 65.7% данных и является статически значимой.

In [71]:
```
x1_data = df[X1].to_numpy().reshape(-1, 1)
x2_data = df[X2].to_numpy().reshape(-1, 1)
y_data = df["Индекс Счастья"].to_numpy()
```

```
X1_ = sm.add_constant(x1_data)
X2_ = sm.add_constant(x2_data)

X = sm.add_constant(df[[X1, X2]])


est = sm.OLS(y_data, X)
est2 = est.fit()

print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.643
Model:                            OLS   Adj. R-squared:                  0.609
Method:                 Least Squares   F-statistic:                     18.93
Date:                Thu, 28 Dec 2023   Prob (F-statistic):           1.99e-05
Time:                        02:27:03   Log-Likelihood:                 13.355
No. Observations:                  24   AIC:                            -20.71
Df Residuals:                      21   BIC:                            -17.18
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=====
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                    0.0611      0.109      0.559      0.582      -0.166
0.289
ВВП на душу населения    0.7322      0.155      4.735      0.000       0.411
1.054
Свобода                  0.2083      0.138      1.506      0.147      -0.079
0.496
==============================================================================
Omnibus:                        3.790   Durbin-Watson:                   0.999
Prob(Omnibus):                  0.150   Jarque-Bera (JB):                2.157
Skew:                          -0.679   Prob(JB):                        0.340
Kurtosis:                       3.559   Cond. No.                         8.18
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Вывод:

1. ВВП на душу населения. Изменил свой коэффицент и стал более стат.значимым.
2. Свобода. Стала менее стат.значимой и так же изменила свой коэффицент. Попробуем его удалить.
3. Сама модель теперь предсказывает примерно 64.3% данных.

In [73]:
```
x1_data = df[X1].to_numpy().reshape(-1, 1)
y_data = df["Индекс Счастья"].to_numpy()
```

```
X1_ = sm.add_constant(x1_data)
X = sm.add_constant(df[X1])


est = sm.OLS(y_data, X)
est2 = est.fit()

print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.605
Model:                            OLS   Adj. R-squared:                  0.587
Method:                 Least Squares   F-statistic:                     33.66
Date:                Thu, 28 Dec 2023   Prob (F-statistic):           7.76e-06
Time:                        02:30:36   Log-Likelihood:                 12.124
No. Observations:                  24   AIC:                            -20.25
Df Residuals:                      22   BIC:                            -17.89
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.1445      0.097      1.489      0.151      -0.057       0.346
ВВП на душу населения     0.8326      0.144      5.802      0.000       0.535       1.130
==============================================================================
Omnibus:                        0.406   Durbin-Watson:                   0.985
Prob(Omnibus):                  0.816   Jarque-Bera (JB):                0.408
Skew:                          -0.266   Prob(JB):                        0.816
Kurtosis:                       2.647   Cond. No.                         6.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Вывод:

1. ВВП на душу населения. Изменил свой коэффицент и стал более стат.значимым.
2. Константа все так же остается не стат. значимой, удалим её и сравним результаты.
3. Модель теперь предсказывает примерно 60.5% данных.

In [74]:
```
x1_data = df[X1].to_numpy().reshape(-1, 1)
y_data = df["Индекс Счастья"].to_numpy()


est = sm.OLS(y_data, x1_data)
```

```
est2 = est.fit()

print(est2.summary())
```

```
                                 OLS Regression Results
================================================================================
===
Dep. Variable:                      y   R-squared (uncentered):                   0.
954
Model:                            OLS   Adj. R-squared (uncentered):              0.
952
Method:                 Least Squares   F-statistic:                              48
0.3
Date:                Thu, 28 Dec 2023   Prob (F-statistic):                    6.52e
-17
Time:                        02:33:48   Log-Likelihood:                          10.
972
No. Observations:                  24   AIC:                                     -1
9.94
Df Residuals:                      23   BIC:                                     -1
8.77
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             1.0350      0.047     21.916      0.000       0.937       1.133
==============================================================================
Omnibus:                        0.665   Durbin-Watson:                   1.381
Prob(Omnibus):                  0.717   Jarque-Bera (JB):                0.283
Skew:                          -0.265   Prob(JB):                        0.868
Kurtosis:                       2.962   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a
constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Вывод:

Теперь остался только коэффиент при параметре ВВП на душу населения и модель предсказывает примерно 95.4% данных, что является отличным результатом. Модель так же статически значима.
Как мы видим, такая модель является лучшей.

2020:

In [212…

```
x123_data = df2[[X1, X2, X3]]
y_data = df2['Индекс счастья']

X_train, X_test, y_train, y_test = train_test_split(x123_data, y_data, test_size=0.
```

In [213…
```
modelx123 = LinearRegression()

modelx123.fit(X_train, y_train)

coeff_df = pd.DataFrame(modelx123.coef_, x123_data.columns, columns=['Coefficient']
coeff_df
```

Out[213…

	Coefficient
ВВП на душу населения	0.249731
Свобода	0.516895
Щедрость	-0.095227

In [214…
```
print('coefficient of determination(R_squared):', modelx123.score(X_train, y_train)
print('intercept:', modelx123.intercept_)
print('slope:', modelx123.coef_[0])
```

```
coefficient of determination(R_squared): 0.7714674374424867
intercept: 0.1396277948740703
slope: 0.24973095568412657
```

In [215…
```
y_pred = modelx123.predict(X_test)
```

In [216…
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred
```

```
Mean Absolute Error: 0.04827537696935991
Mean Squared Error: 0.002330512021533805
Root Mean Squared Error: 0.04827537696935991
```

In [75]:
```
x1_data = df2[X1].to_numpy().reshape(-1, 1)
x2_data = df2[X2].to_numpy().reshape(-1, 1)
x3_data = df2[X3].to_numpy().reshape(-1, 1)
y_data = df2["Индекс счастья"].to_numpy()

X1_ = sm.add_constant(x1_data)
X2_ = sm.add_constant(x2_data)
X3_ = sm.add_constant(x3_data)


X = sm.add_constant(df2[[X1, X2, X3]])


est = sm.OLS(y_data, X)
est2 = est.fit()

print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.730
Method:                 Least Squares   F-statistic:                     19.05
Date:                Thu, 28 Dec 2023   Prob (F-statistic):           1.12e-05
Time:                        02:44:13   Log-Likelihood:                 20.604
No. Observations:                  21   AIC:                            -33.21
Df Residuals:                      17   BIC:                            -29.03
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1686      0.133      1.265      0.223      -0.113       0.450
ВВП на душу населения     0.2139      0.137      1.560      0.137      -0.075       0.503
Свобода        0.5161      0.113      4.560      0.000       0.277       0.755
Щедрость      -0.1285      0.147     -0.873      0.395      -0.439       0.182
==============================================================================
Omnibus:                        0.468   Durbin-Watson:                   1.129
Prob(Omnibus):                  0.791   Jarque-Bera (JB):                0.561
Skew:                           0.084   Prob(JB):                        0.755
Kurtosis:                       2.217   Cond. No.                         14.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Уберём коэффицент при параметре X3, щедрость, и посмотрим, как поведет себя модель в таком случае. Пока что модель предсказывает 77% данных и является статически значимой. Удаляем именно этот коэффицент, т.к. его стат.значимость наименьшая.

In [81]:
```
x1_data = df2[X1].to_numpy().reshape(-1, 1)
x2_data = df2[X2].to_numpy().reshape(-1, 1)
y_data = df2["Индекс счастья"].to_numpy()

X1_ = sm.add_constant(x1_data)
X2_ = sm.add_constant(x2_data)


X = sm.add_constant(df2[[X1, X2]])


est = sm.OLS(y_data, X)
est2 = est.fit()
```

```
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.761
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     28.58
Date:                Thu, 28 Dec 2023   Prob (F-statistic):           2.59e-06
Time:                        03:05:35   Log-Likelihood:                 20.143
No. Observations:                  21   AIC:                            -34.29
Df Residuals:                      18   BIC:                            -31.15
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0767      0.081      0.946      0.357      -0.094       0.247
ВВП на душу населения     0.2862      0.108      2.639      0.017       0.058       0.514
Свобода          0.5396      0.109      4.943      0.000       0.310       0.769
==============================================================================
Omnibus:                        0.511   Durbin-Watson:                   1.296
Prob(Omnibus):                  0.775   Jarque-Bera (JB):                0.574
Skew:                          -0.012   Prob(JB):                        0.751
Kurtosis:                       2.190   Cond. No.                         8.65
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Модель предсказывает 76% данных. Все коэффициенты при переменных стат.значимы, поэтому попробуем убрать свободный член.

In [82]:
```
x1_data = df2[X1].to_numpy().reshape(-1, 1)
x2_data = df2[X2].to_numpy().reshape(-1, 1)
y_data = df2["Индекс счастья"].to_numpy()

X = df2[[X1, X2]]

est = sm.OLS(y_data, X)

est2 = est.fit()

print(est2.summary())
```

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.981
Model:	OLS	Adj. R-squared (uncentered):	0.979
Method:	Least Squares	F-statistic:	493.6
Date:	Thu, 28 Dec 2023	Prob (F-statistic):	4.19e-17
Time:	03:05:42	Log-Likelihood:	19.634
No. Observations:	21	AIC:	-35.27
Df Residuals:	19	BIC:	-33.18
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
ВВП на душу населения	0.3283	0.099	3.329	0.004	0.122	0.535
Свобода	0.5981	0.090	6.667	0.000	0.410	0.786

Omnibus:	0.643	Durbin-Watson:	1.528
Prob(Omnibus):	0.725	Jarque-Bera (JB):	0.660
Skew:	-0.153	Prob(JB):	0.719
Kurtosis:	2.187	Cond. No.	6.23

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

ВЫВОД:

Такая модель предсказывает примерно 98.1% данных, что является отличным результатом. Обе переменные ВВП на душу населения и Свобода являются стат.значимыми. Сама модель так же является стат.значимой.

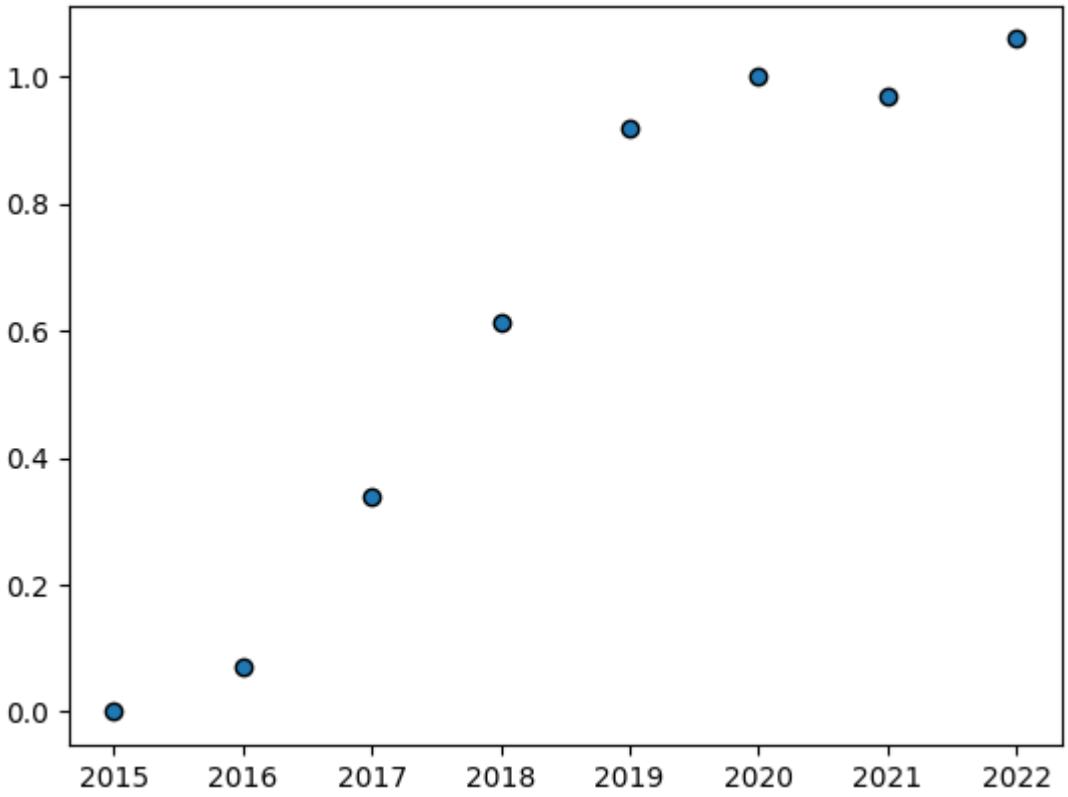
Задание №8

Для анализа возьмём уровень счастья Гондураса. С начала соберем данные с каждого года и нормируем их. Построим график, чтобы посмотреть, что у нас получилось.

In [160…
```
happiness_index = {}
```

In [189…
```
mmin=10000
mmax = 0

for i in range(2015, 2018):
    buff = pd.read_excel(r"C:\Users\Tezer\Downloads\idz_data.xlsx", sheet_name=str(
    buff = buff[buff["Country"]=="Honduras"]["Индекс Счастья"]
    happiness_index[i]=(buff.iloc(0)[0])
    mmin = min(mmin, buff.iloc(0)[0])
    mmax = max(mmax, buff.iloc(0)[0])
for i in range(2018, 2020):
    buff = pd.read_excel(r"C:\Users\Tezer\Downloads\idz_data.xlsx", sheet_name=str(
    buff = buff[buff["Country or region"]=="Honduras"]["Индекс Счастья"]
    happiness_index[i]=(buff.iloc(0)[0])
    mmin = min(mmin, buff.iloc(0)[0])
    mmax = max(mmax, buff.iloc(0)[0])
for i in range(2020, 2022):
    buff = pd.read_excel(r"C:\Users\Tezer\Downloads\idz_data.xlsx", sheet_name=str(
    buff = buff[buff["Country name"]=="Honduras"]["Индекс счастья"]
    happiness_index[i]=(buff.iloc(0)[0])
    mmin = min(mmin, buff.iloc(0)[0])
    mmax = max(mmax, buff.iloc(0)[0])

happiness_index["2022"]=6.022
delta = mmax-mmin
for i in happiness_index.keys():
    happiness_index[i] = (happiness_index[i]-mmin)/delta
plt.scatter(happiness_index.keys(),happiness_index.values(), edgecolor='black')
```

Out[189…
```
<matplotlib.collections.PathCollection at 0x259796ce6a0>
```

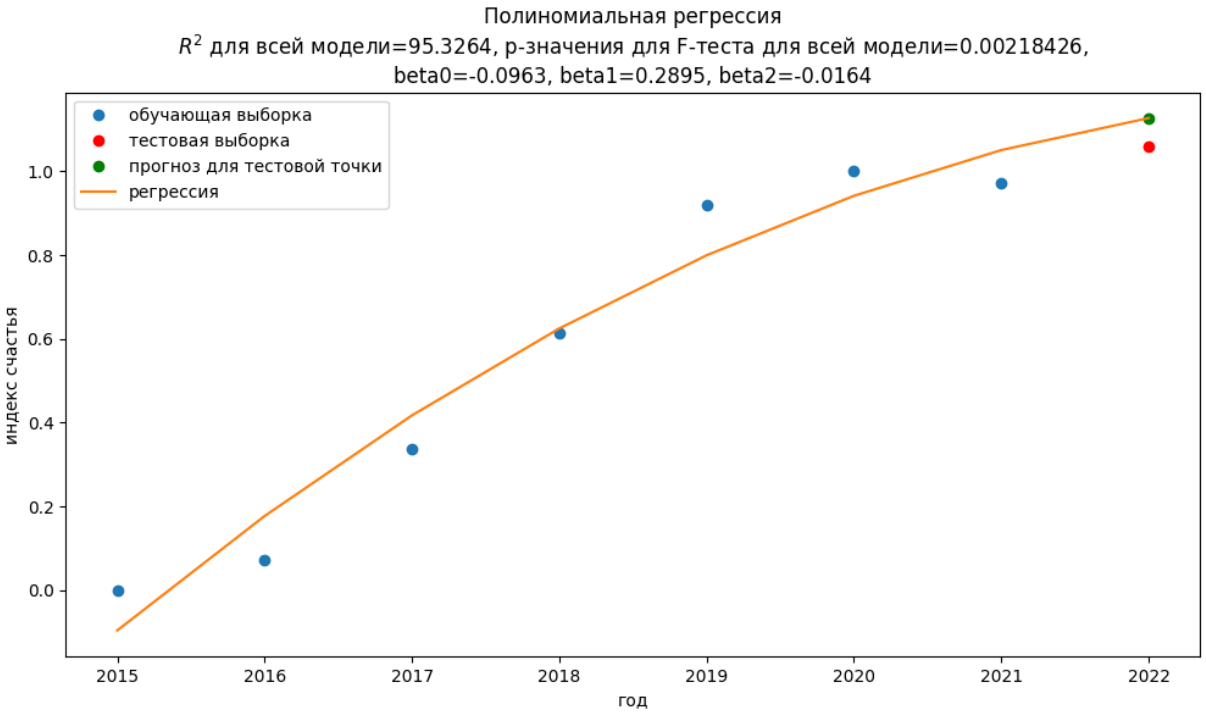


Обучим регрессионную модель и посмотрим на её предсказание.

In [190…

```
x = np.arange(0,8)
y = happiness_index.values()
y = list(y)
x_train = x[:-1].reshape(-1,1)
y_train = y[:-1]
degree = 2
poly = PolynomialFeatures(degree=degree)
x_poly_train = poly.fit_transform(x_train)
model = LinearRegression()
model.fit(x_poly_train, y_train)

x_test = x[-1].reshape(-1,1)
x_poly_test = poly.fit_transform(x_test)
y_pred_test = model.predict(x_poly_test)
y_pred = model.predict(poly.fit_transform(x.reshape(-1,1)))
mod = sm.OLS(y_train, x_poly_train)
fii = mod.fit()
r2 = np.round(r2_score(y_train, y_pred[:-1])*100,4)
p_val = np.round(fii.f_pvalue, 8)
plt.figure(figsize=(12,6))
plt.plot(x[:-1], y_train, 'o', label='обучающая выборка')
plt.plot(x[-1], y[-1], 'ro', label='тестовая выборка')
plt.plot(x[-1], y_pred_test, 'go', label='прогноз для тестовой точки')
plt.plot(x, y_pred, label='регрессия')
plt.legend()
plt.xticks(ticks=list(range(8)), labels=[str(i) for i in
range(2015,2023)])
plt.xlabel('год')
```

```
plt.ylabel('индекс счастья')
plt.title('''Полиномиальная регрессия
$R^2$ для всей модели=%s, p-значения для F-теста для всей модели=%s,
beta0=%s, beta1=%s, beta2=%s'''%(r2, p_val,
np.round(model.intercept_,4), np.round(model.coef_[1],4),
np.round(model.coef_[2],4)))
plt.show()
```



In [ ]:

In [ ]:

In [ ]: