Project -1 :

Which Debts Are Worth the Bank's Effort ?

Project Description :

After a debt has been legally declared "uncollectable" by a bank, the account is considered to be "charged-off." But that doesn't mean the bank simply walks away from the debt. They still want to collect some of the money they are owed. In this project, you will look at a situation where a bank assigned delinquent customers to different recovery strategies based on the expected amount the bank believed it would recover from the customer. The goal for the data scientist is to determine in this non-random assignment whether the incremental amount the bank earns exceeded the additional cost of assigning customers to a higher recovery strategy. Threshold assignments like this also one occur in medicine (above a certain temperature you get medicine), education (above a certain test score students get admitted to a special class), other areas of finance (above a certain wealth customers get different levels of service), and public sector (below a certain income someone is eligible for housing benefits). Regression discontinuity is an intuitive and useful analysis method in any situation of a threshold assignment.

Project Tasks:

- Regression discontinuity: banking recovery :

After a debt has been legally declared "uncollectable" by a bank, the account is considered to be "charged-off." But that doesn't mean the bank simply walks away from the debt. They still want to collect some of the money they are owed. The bank will score the account to assess the expected recovery amount, that is, the expected amount that the bank may be able to receive from the customer in the future (for a fixed time period such as one year). This amount is a function of the probability of the customer paying, the total debt, and other factors that impact the ability and willingness to pay. The bank has implemented different recovery strategies at different thresholds (1000,2000, etc.) where the greater the expected recovery amount, the more effort the bank puts into contacting the customer. For low recovery amounts (Level 0), the bank just adds the customer's contact information to their automatic dialer and emailing system. For higher recovery strategies, the bank incurs more costs as they leverage human resources in more efforts to contact the customer and obtain payments. Each additional level of recovery strategy requires an additional $50 per customer so that customers in the Recovery Strategy Level 1 cost the company$50 more than those in Level 0. Customers in Level 2 cost $50 more than those in Level 1, etc. The big question: does the extra amount that is recovered at the higher strategy level exceed the extra$50 in costs? In other words, was there a jump (also called a "discontinuity") of more than $50 in the amount recovered at the higher strategy level?

In [1]:
```python
# Import the modules
import pandas as pd
import numpy as np

# Read in dataset
df = pd.read_csv(r"C:\Users\user\Downloads\bank_data.csv")

# Print the first few rows of the DataFrame
df.head()
```

Out[1]:

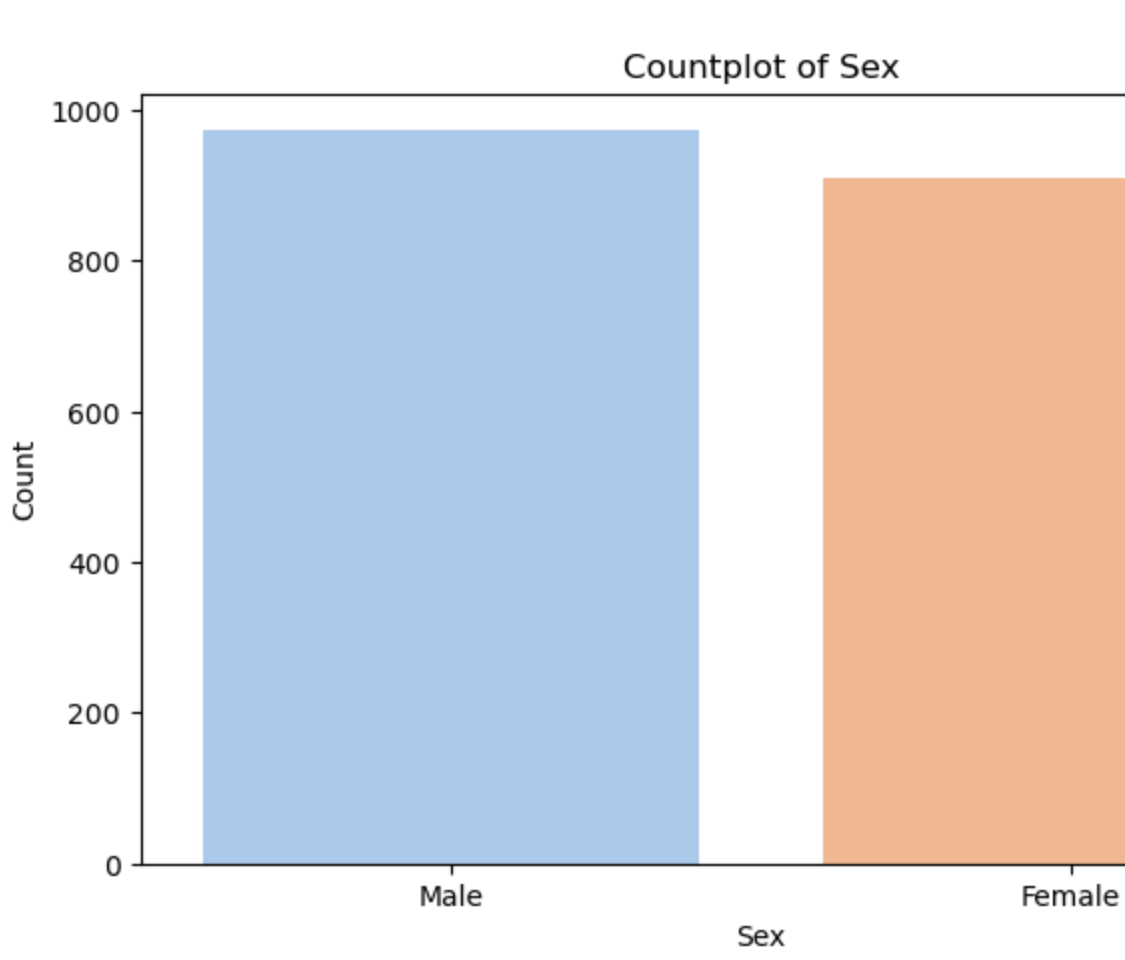
	id	expected_recovery_amount	actual_recovery_amount	recovery_strategy	age	sex
0	2030	194	263.540	Level 0 Recovery	19	Male
1	1150	486	416.090	Level 0 Recovery	25	Female
2	380	527	429.350	Level 0 Recovery	27	Male
3	1838	536	296.990	Level 0 Recovery	25	Male
4	1995	541	346.385	Level 0 Recovery	34	Male

Explanation: This code imports the pandas and numpy libraries, reads a CSV file ("bank_data.csv") into a DataFrame (`df`), and displays the first few rows of the DataFrame using the `head()` function. The dataset likely contains banking-related information, and this code serves as an initial exploration of the data.

- Graphical exploratory data analysis :

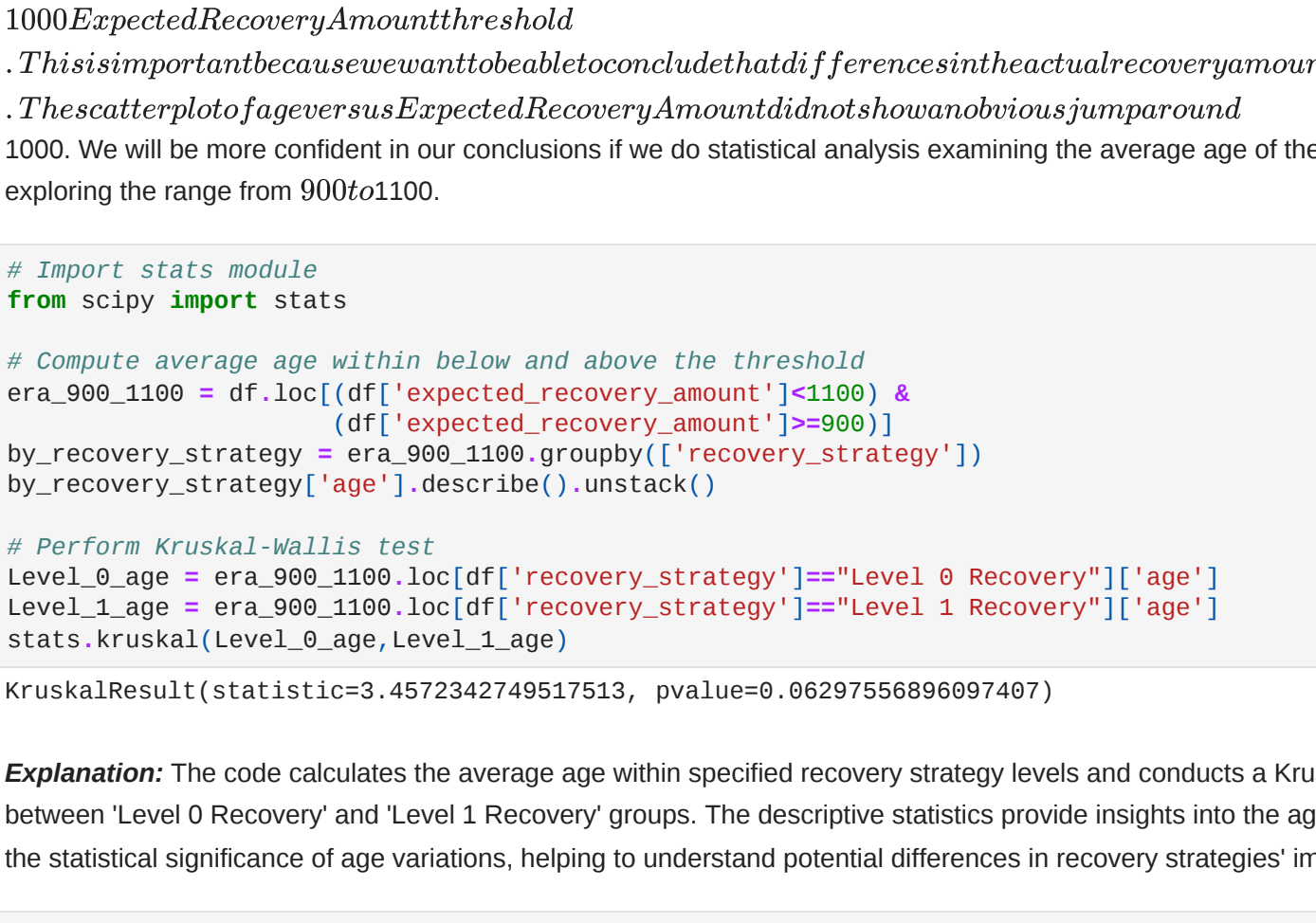
The bank has implemented different recovery strategies at different thresholds (1000,2000, $3000 and$5000) where the greater the Expected Recovery Amount, the more effort the bank puts into contacting the customer. Zeroing in on the first transition (between Level 0 and Level 1) means we are focused on the population with Expected Recovery Amounts between $0 and$2000 where the transition between Levels occurred at $1000. We know that the customers in Level 1 (expected recovery amounts between$1001 and $2000) received more attention from the bank and, by definition, they had higher Expected Recovery Amounts than the customers in Level 0 (between$1 and $1000). Here's a quick summary of the Levels and thresholds again: Level 0: Expected recovery amounts >$0 and <=$1000 Level 1: Expected recovery amounts >$1000 and <=$2000 The threshold of$1000 separates Level 0 from Level 1 A key question is whether there are other factors besides Expected Recovery Amount that also varied systematically across the $1000 threshold. For example, does the customer age show a jump (discontinuity) at the$1000 threshold or does that age vary smoothly? We can examine this by first making a scatter plot of the age as a function of Expected Recovery Amount for a small window of Expected Recovery Amount, $0 to$2000. This range covers Levels 0 and 1.

In [2]:
```python
# Scatter plot of Age vs. Expected Recovery Amount
from matplotlib import pyplot as plt
%matplotlib inline
plt.scatter(x=df['expected_recovery_amount'], y=df['age'], c="b", s=2)
plt.xlim(0, 2000)
plt.ylim(0, 60)
plt.xlabel("Expected Recovery Amount")
plt.ylabel("Age")
plt.show()
```



Explanation: This code generates a scatter plot using matplotlib, where the x-axis represents "Expected Recovery Amount," the y-axis represents "Age," and each data point is visually represented by a blue dot. The scatter plot is limited to the range of 0 to 2000 on the x-axis and 0 to 60 on the y-axis. The code aims to explore the relationship between age and the expected recovery amount in the provided dataset.

In [3]:
```python
plt.figure(figsize=(8, 5))
sns.countplot(x='sex', data=df, palette='pastel')
plt.title('Countplot of Sex')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()
```



Explanation: This code uses seaborn (`sns`) to create a countplot, visualizing the distribution of the 'sex' variable in the DataFrame (`df`). The countplot displays the number of occurrences for each category in the 'sex' column. The plot is styled with a pastel color palette and includes a title, x-axis label ('Sex'), and y-axis label ('Count'). The `plt.figure(figsize=(8, 5))` sets the figure size to 8 by 5 inches. Overall, the code provides a visual summary of the distribution of sexes in the dataset.

- Statistical test: age vs. expected recovery amount :

We want to convince ourselves that variables such as age and sex are similar above and below the $1000 Expected Recovery Amount threshold. This is important because we want to be able to conclude that differences in the actual recovery amount are due to the higher Recovery Strategy and not due to some other di... The scatter plot of age versus Expected Recovery Amount did not show an obvious jump around$1000. We will be more confident in our conclusions if we do statistical analysis examining the average age of the customers just above and just below the threshold. We can start by exploring the range from $900 to$1100.

In [4]:
```python
# Import stats module
from scipy import stats

# Compute average age within below and above the threshold
era_900_1100 = df.loc[(df['expected_recovery_amount']<1100) &
                      (df['expected_recovery_amount']>=900)]
by_recovery_strategy = era_900_1100.groupby(['recovery_strategy'])
by_recovery_strategy['age'].describe().unstack()

# Perform Kruskal-Wallis test
Level_0_age = era_900_1100.loc[df['recovery_strategy']=="Level 0 Recovery"]['age']
Level_1_age = era_900_1100.loc[df['recovery_strategy']=="Level 1 Recovery"]['age']
stats.kruskal(Level_0_age,Level_1_age)
```

Out[4]:
```
KruskalResult(statistic=3.4572342749517513, pvalue=0.06297556896097407)
```

Explanation: The code calculates the average age within specified recovery strategy levels and conducts a Kruskal-Wallis test to assess if there are significant age differences between 'Level 0 Recovery' and 'Level 1 Recovery' groups. The descriptive statistics provide insights into the age distribution within each group, while the Kruskal-Wallis test evaluates the statistical significance of age variations, helping to understand potential differences in recovery strategies' impact on customer age.

In [5]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a FacetGrid with histograms for each recovery strategy
g = sns.FacetGrid(era_900_1100, col="recovery_strategy", height=6)
g.map(plt.hist, 'age', bins=20, color='skyblue', edgecolor='black')

# Set labels and title
g.set_axis_labels("Age", "Frequency")
g.set_titles("Histogram of Age Distribution for {col_name}")

# Show the plots
plt.show()
```

Explanation: This code utilizes Seaborn to create a FacetGrid with histograms for the 'age' variable, distinguishing between 'Level 0 Recovery' and 'Level 1 Recovery' groups within the specified recovery strategy window (Expected Recovery Amount between 900 and 1100). Each histogram is color-coded with sky blue and displays 20 bins with black edges. The resulting plots provide a visual representation of the age distribution for each recovery strategy, aiding in the exploration of potential differences in age profiles.

- Statistical test: sex vs. expected recovery amount :

We were able to convince ourselves that there is no major jump in the average customer age just above and just below the $1000 threshold by doing a statistical test as well as exploring it graphically with a scatter plot. We want to also test that the percentage of customers that are male does not jump as well across the$1000 threshold. We can start by exploring the range of $900 to$1100 and later adjust this range.

In [6]:
```python
# Number of customers in each category
crosstab = pd.crosstab(df.loc[(df['expected_recovery_amount']<1100) &
                              (df['expected_recovery_amount']>=900)]['recovery_strategy'],
                       df['sex'])
print(crosstab)

# Chi-square test
chi2_stat, p_val, dof, ex = stats.chi2_contingency(crosstab)
print(p_val)
```

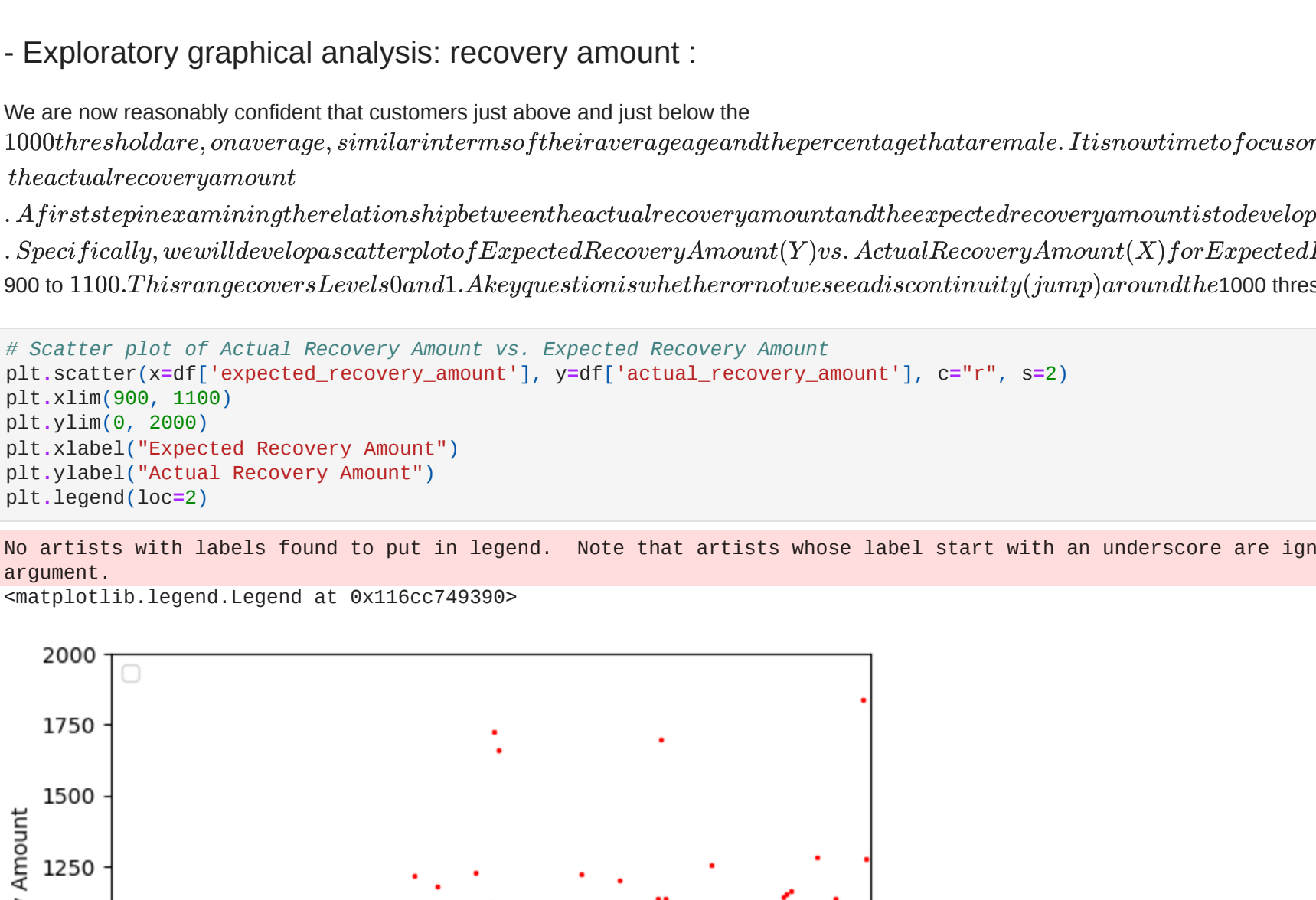
```
sex                Female  Male
recovery_strategy
Level 0 Recovery       32    57
Level 1 Recovery       39    55
0.5377947810444592
```

Explanation: This code first creates a cross-tabulation (`crosstab`) between 'recovery_strategy' and 'sex' for customers within the specified expected recovery amount window (between 900 and 1100). It then performs a chi-square test (`chi2_contingency`) on the cross-tabulated data to assess whether there is a significant association between recovery strategy and gender. The resulting p-value (`p_val`) indicates the likelihood of observing the observed distribution under the assumption of independence between recovery strategy and gender. If the p-value is low, it suggests a significant association between the two variables.

In [7]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Filter data within the specified range
filtered_df = df[(df['expected_recovery_amount'] >= 900) & (df['expected_recovery_amount'] <= 1100)]

# Box plot of Actual Recovery Amount by Expected Recovery Amount
plt.figure(figsize=(10, 6))
sns.boxplot(x='expected_recovery_amount', y='actual_recovery_amount', data=filtered_df, color='skyblue')
plt.xlabel("Expected Recovery Amount")
plt.ylabel("Actual Recovery Amount")
plt.title("Box Plot of Actual Recovery Amount by Expected Recovery Amount")
plt.show()
```



Explanation: This code filters the DataFrame to include only rows with an 'expected_recovery_amount' between 900 and 1100 and then generates a box plot using Seaborn. The box plot visualizes the distribution of 'actual_recovery_amount' for each level of 'expected_recovery_amount.' The x-axis represents the expected recovery amount, and the y-axis represents the actual recovery amount. The plot is styled with a sky blue color, providing a graphical summary of the relationship between these two variables.

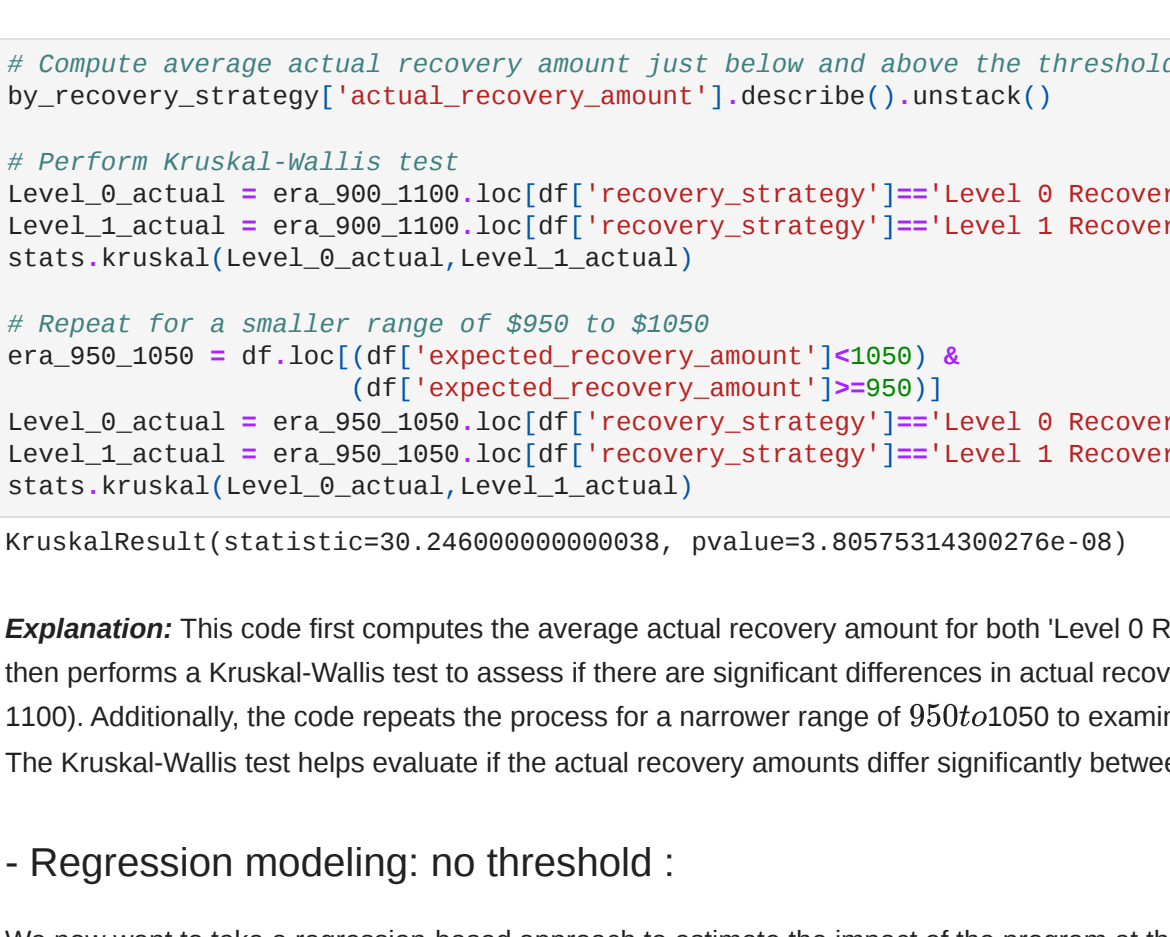
- Exploratory graphical analysis: recovery amount :

We are now reasonably confident that customers just above and just below the $1000 threshold are, on average, similar in terms of their average age and the percentage that are male. It is now time to focus on the key outcome of interest, the actual recovery amount. A first step in examining the relationship between the actual recovery amount and the expected recovery amount is to develop a scatter plot where we want to focus our atte... Specifically, we will develop a scatter plot of Expected Recovery Amount (Y) vs. Actual Recovery Amount (X) for Expected Recovery Amounts between$900 to $1100. This range covers Levels 0 and 1. A key question is whether or not we see a discontinuity (jump) around the$1000 threshold.

In [8]:
```python
# Scatter plot of Actual Recovery Amount vs. Expected Recovery Amount
plt.scatter(x=df['expected_recovery_amount'], y=df['actual_recovery_amount'], c="r", s=2)
plt.xlim(900, 1100)
plt.ylim(0, 2000)
plt.xlabel("Expected Recovery Amount")
plt.ylabel("Actual Recovery Amount")
plt.legend(loc=2)
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```

Out[8]:
```
<matplotlib.legend.Legend at 0x116cc749390>
```



Explanation: This code generates a scatter plot using matplotlib, where the x-axis represents "Expected Recovery Amount," the y-axis represents "Actual Recovery Amount," and each data point is represented by a red dot. The plot is limited to the range of 900 to 1100 on the x-axis and 0 to 2000 on the y-axis. The legend is placed at the top-left corner (loc=2). This scatter plot visualizes the relationship between the expected and actual recovery amounts, providing insights into their correlation within the specified range.

- Statistical analysis: recovery amount :

Just as we did with age, we can perform statistical tests to see if the actual recovery amount has a discontinuity above the $1000 threshold. We are going to do this for two different windows of the expected recovery amount$900 to $1100 and for a narrow range of$950 to $1050 to see if our results are consistent. Again, the statistical test we will use is the Kruskal-Wallis test, a test that makes no assumptions about the distribution of the actual recovery amount.

In [9]:
```python
# Compute average actual recovery amount just below and above the threshold
by_recovery_strategy['actual_recovery_amount'].describe().unstack()

# Perform Kruskal-Wallis test
Level_0_actual = era_900_1100.loc[df['recovery_strategy']=='Level 0 Recovery']['actual_recovery_amount']
Level_1_actual = era_900_1100.loc[df['recovery_strategy']=='Level 1 Recovery']['actual_recovery_amount']
stats.kruskal(Level_0_actual,Level_1_actual)

# Repeat for a smaller range of $950 to $1050
era_950_1050 = df.loc[(df['expected_recovery_amount']<1050) &
                      (df['expected_recovery_amount']>=950)]
Level_0_actual = era_950_1050.loc[df['recovery_strategy']=='Level 0 Recovery']['actual_recovery_amount']
Level_1_actual = era_950_1050.loc[df['recovery_strategy']=='Level 1 Recovery']['actual_recovery_amount']
stats.kruskal(Level_0_actual,Level_1_actual)
```

Out[9]:
```
KruskalResult(statistic=30.246000000000038, pvalue=3.80575314300276e-08)
```

Explanation: This code first computes the average actual recovery amount for both 'Level 0 Recovery' and 'Level 1 Recovery' groups, providing descriptive statistics for each group. It then performs a Kruskal-Wallis test to assess if there are significant differences in actual recovery amounts between these two recovery strategy levels for the initial range ($900 to$1100). Additionally, the code repeats the process for a narrower range of $950 to$1050 to examine consistency in the results across different windows of expected recovery amounts. The Kruskal-Wallis test helps evaluate if the actual recovery amounts differ significantly between the two recovery strategy levels within each specified range.

- Regression modeling: no threshold :

We now want to take a regression-based approach to estimate the impact of the program at the $1000 threshold using the data that is just above and just below the threshold. In order to do that, we will build two models. The first model does not have a threshold while the second model will include a threshold. The first model predicts the actual recovery amount (outcome or dependent variable) as a function of the expected recovery amount (input or independent variable). We expect that there will be a strong positive relationship between these two variables.

In [10]:
```python
# Import stats models
import statsmodels.api as sm

# Define X and y
X = era_900_1100['expected_recovery_amount']
y = era_900_1100['actual_recovery_amount']
X = sm.add_constant(X)

# Build linear regression model
model = sm.OLS(y, X).fit()
predictions = model.predict(X)

# Print out the model summary statistics
model.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	actual_recovery_amount	**R-squared:**	0.261
Model:	OLS	**Adj. R-squared:**	0.256
Method:	Least Squares	**F-statistic:**	63.78
Date:	Tue, 23 Jan 2024	**Prob (F-statistic):**	1.56e-13
Time:	10:55:35	**Log-Likelihood:**	-1278.9
No. Observations:	183	**AIC:**	2562.
Df Residuals:	181	**BIC:**	2568.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1978.7597	347.741	-5.690	0.000	-2664.907	-1292.612
expected_recovery_amount	2.7577	0.345	7.986	0.000	2.076	3.439

Omnibus:	64.493	**Durbin-Watson:**	1.777
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	185.818
Skew:	1.463	**Prob(JB):**	4.47e-41
Kurtosis:	6.977	**Cond. No.**	1.80e+04

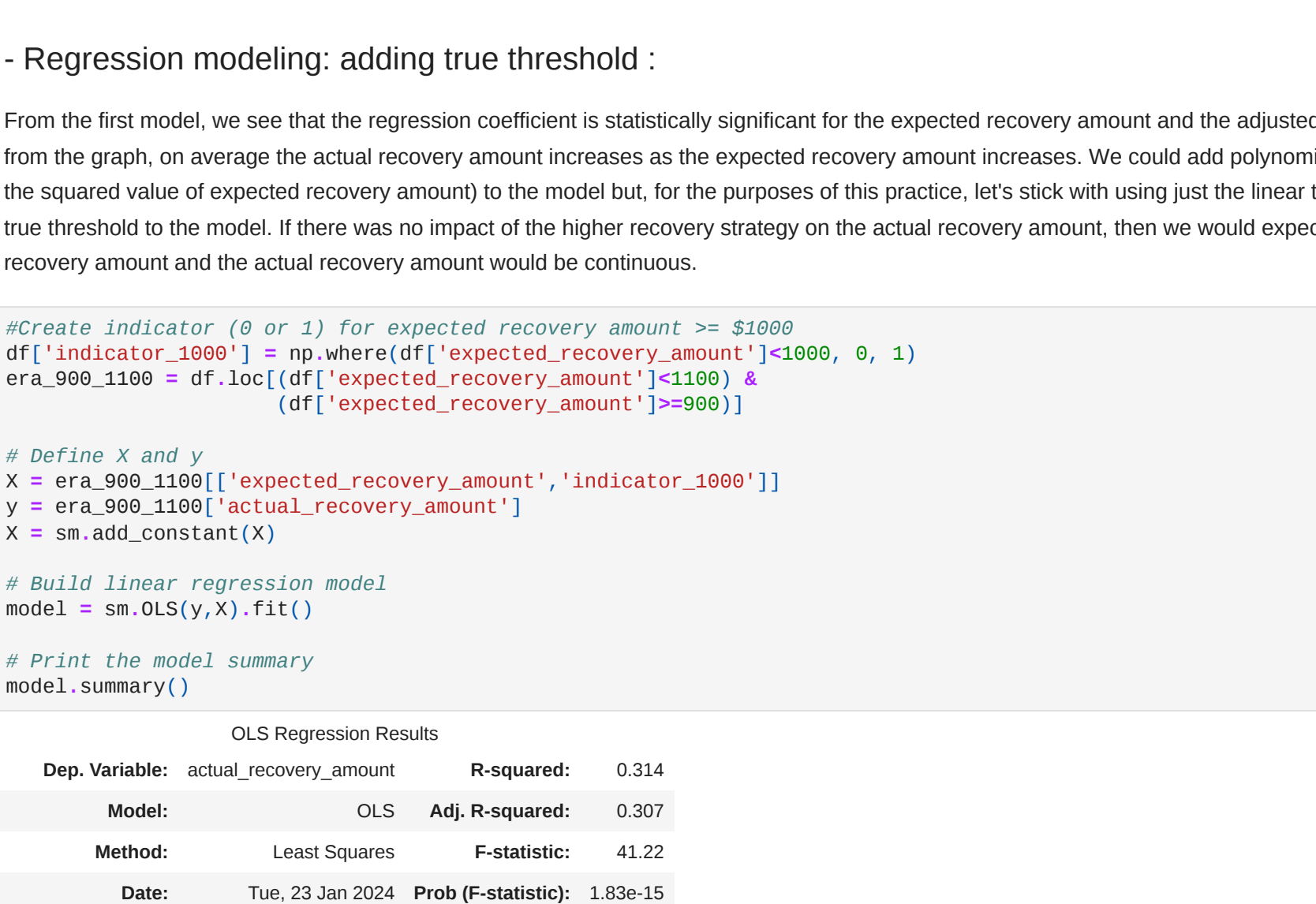
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.8e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Explanation: This code utilizes the statsmodels library to perform a simple linear regression. It defines the independent variable ('X') as the 'expected_recovery_amount' and the dependent variable ('y') as the 'actual_recovery_amount' for the specified range ($900 to$1100). The code adds a constant term to the independent variable, fits a linear regression model using Ordinary Least Squares (OLS), and prints out the summary statistics of the regression model, including coefficients, R-squared, and other relevant information. The model aims to estimate the impact of the expected recovery amount on the actual recovery amount within the specified range.

In [11]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate residuals
residuals = y - predictions

# Create a histogram of residuals
plt.figure(figsize=(10, 6))
sns.histplot(residuals, bins=20, kde=True, color='skyblue')
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.title("Histogram of Residuals")
plt.show()
```



Explanation: This code uses seaborn and matplotlib to calculate the residuals (the differences between the observed and predicted values) from the linear regression model. It then creates a histogram of these residuals with 20 bins and overlays a kernel density estimate (KDE) for a smoother representation. The plot is presented in a figure with a size of 10 by 6 inches, and it is titled "Histogram of Residuals." The histogram helps visualize the distribution of errors in the regression model, providing insights into its performance and any patterns in the residuals.

- Regression modeling: adding true threshold :

From the first model, we see that the regression coefficient is statistically significant for the expected recovery amount and the adjusted R-squared value was about 0.26. As we saw from the graph, on average the actual recovery amount increases as the expected recovery amount increases. We could add polynomial terms of expected recovery amount (such as the squared value of expected recovery amount) to the model but, for the purposes of this practice, let's stick with using just the linear term. The second model adds an indicator of the true threshold to the model. If there was no impact of the higher recovery strategy on the actual recovery amount, then we would expect that the relationship between the expected recovery amount and the actual recovery amount would be continuous.

In [12]:
```python
#Create indicator (0 or 1) for expected recovery amount >= $1000
df['indicator_1000'] = np.where(df['expected_recovery_amount']<1000, 0, 1)
era_900_1100 = df.loc[(df['expected_recovery_amount']<1100) &
                      (df['expected_recovery_amount']>=900)]

# Define X and y
X = era_900_1100[['expected_recovery_amount','indicator_1000']]
y = era_900_1100['actual_recovery_amount']
X = sm.add_constant(X)

# Build linear regression model
model = sm.OLS(y,X).fit()

# Print the model summary
model.summary()
```

Out[12]:

OLS Regression Results

Dep. Variable:	actual_recovery_amount	**R-squared:**	0.314
Model:	OLS	**Adj. R-squared:**	0.307
Method:	Least Squares	**F-statistic:**	41.22
Date:	Tue, 23 Jan 2024	**Prob (F-statistic):**	1.83e-15
Time:	10:55:35	**Log-Likelihood:**	-1272.0
No. Observations:	183	**AIC:**	2550.
Df Residuals:	180	**BIC:**	2560.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.3440	626.274	0.005	0.996	-1232.440	1239.128
expected_recovery_amount	0.6430	0.655	0.981	0.328	-0.650	1.936
indicator_1000	277.6344	74.043	3.750	0.000	131.530	423.739

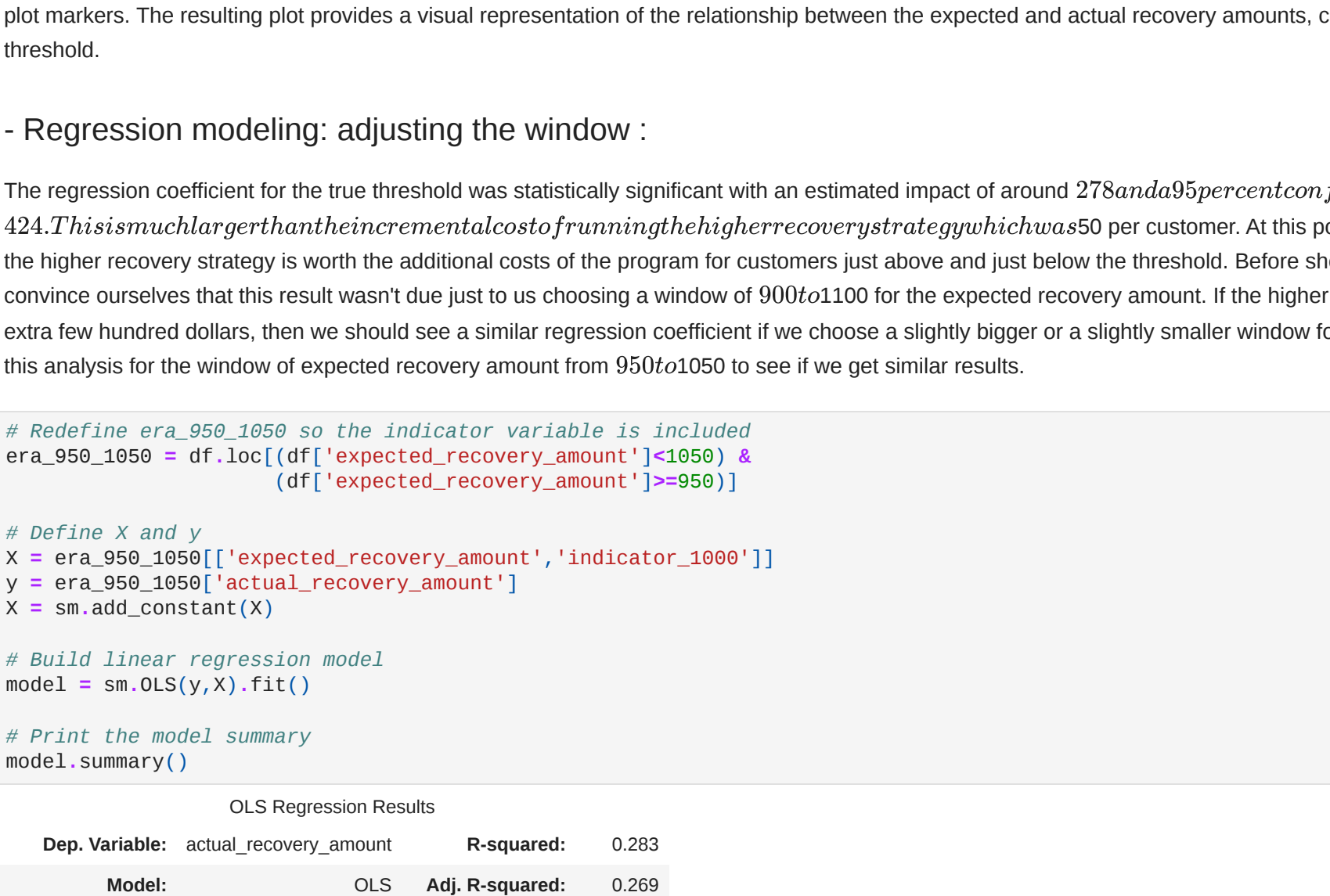
Omnibus:	65.977	**Durbin-Watson:**	1.906
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	186.537
Skew:	1.510	**Prob(JB):**	3.12e-41
Kurtosis:	6.917	**Cond. No.**	3.37e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.37e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Explanation This code creates a binary indicator variable, 'indicator_1000', which takes the value 1 if the 'expected_recovery_amount' is greater than or equal to $1000 and 0 otherwise. It then filters the data to include only records within the range of$900 to $1100. The code defines the independent variables ('X') as both the 'expected_recovery_amount' and the binary indicator, and the dependent variable ('y') as the 'actual_recovery_amount'. A constant term is added to the independent variables, and a linear regression model is built using Ordinary Least Squares (OLS). Finally, the summary statistics of the regression model are printed. This model aims to estimate the impact of the expected recovery amount, taking into account the threshold indicator, on the actual recovery amount within the specified range.

In [15]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Scatter plot with regression line
plt.figure(figsize=(10, 6))
sns.regplot(x='expected_recovery_amount', y='actual_recovery_amount', data=era_900_1100, ci=None, scatter_kws={'s': 20})
plt.xlabel("Expected Recovery Amount")
plt.ylabel("Actual Recovery Amount")
plt.title("Regression Plot with Indicator Variable")
plt.show()
```



Explanation This code generates a scatter plot with a regression line using the Seaborn library. The x-axis represents the 'expected_recovery_amount', and the y-axis represents the 'actual_recovery_amount'. Each point in the scatter plot corresponds to a data point in the 'era_900_1100' dataset. The regression line represents the linear relationship between the expected and actual recovery amounts. The 'ci=None' parameter removes the confidence interval around the regression line, and 'scatter_kws={'s': 20}' adjusts the size of the scatter plot markers. The resulting plot provides a visual representation of the relationship between the expected and actual recovery amounts, considering the indicator variable for the threshold.

- Regression modeling: adjusting the window :

The regression coefficient for the true threshold was statistically significant with an estimated impact of around $278 and a 95 percent confidence interval of$132 to $424. This is much larger than the incremental cost of running the higher recovery strategy which was$50 per customer. At this point, we are feeling reasonably confident that the higher recovery strategy is worth the additional costs of the program for customers just above and just below the threshold. Before showing this to our managers, we want to convince ourselves that this result wasn't due just to us choosing a window of $900 to$1100 for the expected recovery amount. If the higher recovery strategy really had an impact of an extra few hundred dollars, then we should see a similar regression coefficient if we choose a slightly bigger or a slightly smaller window for the expected recovery amount. Let's repeat this analysis for the window of expected recovery amount from $950 to$1050 to see if we get similar results.

In [13]:
```python
# Redefine era_950_1050 so the indicator variable is included
era_950_1050 = df.loc[(df['expected_recovery_amount']<1050) &
                      (df['expected_recovery_amount']>=950)]

# Define X and y
X = era_950_1050[['expected_recovery_amount','indicator_1000']]
y = era_950_1050['actual_recovery_amount']
X = sm.add_constant(X)

# Build linear regression model
model = sm.OLS(y,X).fit()

# Print the model summary
model.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	actual_recovery_amount	**R-squared:**	0.283
Model:	OLS	**Adj. R-squared:**	0.269
Method:	Least Squares	**F-statistic:**	18.99
Date:	Tue, 23 Jan 2024	**Prob (F-statistic):**	1.12e-07
Time:	10:55:35	**Log-Likelihood:**	-692.92
No. Observations:	99	**AIC:**	1392.
Df Residuals:	96	**BIC:**	1400.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-279.5243	1840.707	-0.152	0.880	-3933.298	3374.250
expected_recovery_amount	0.9189	1.886	0.487	0.627	-2.825	4.663
indicator_1000	286.5337	111.352	2.573	0.012	65.502	507.566

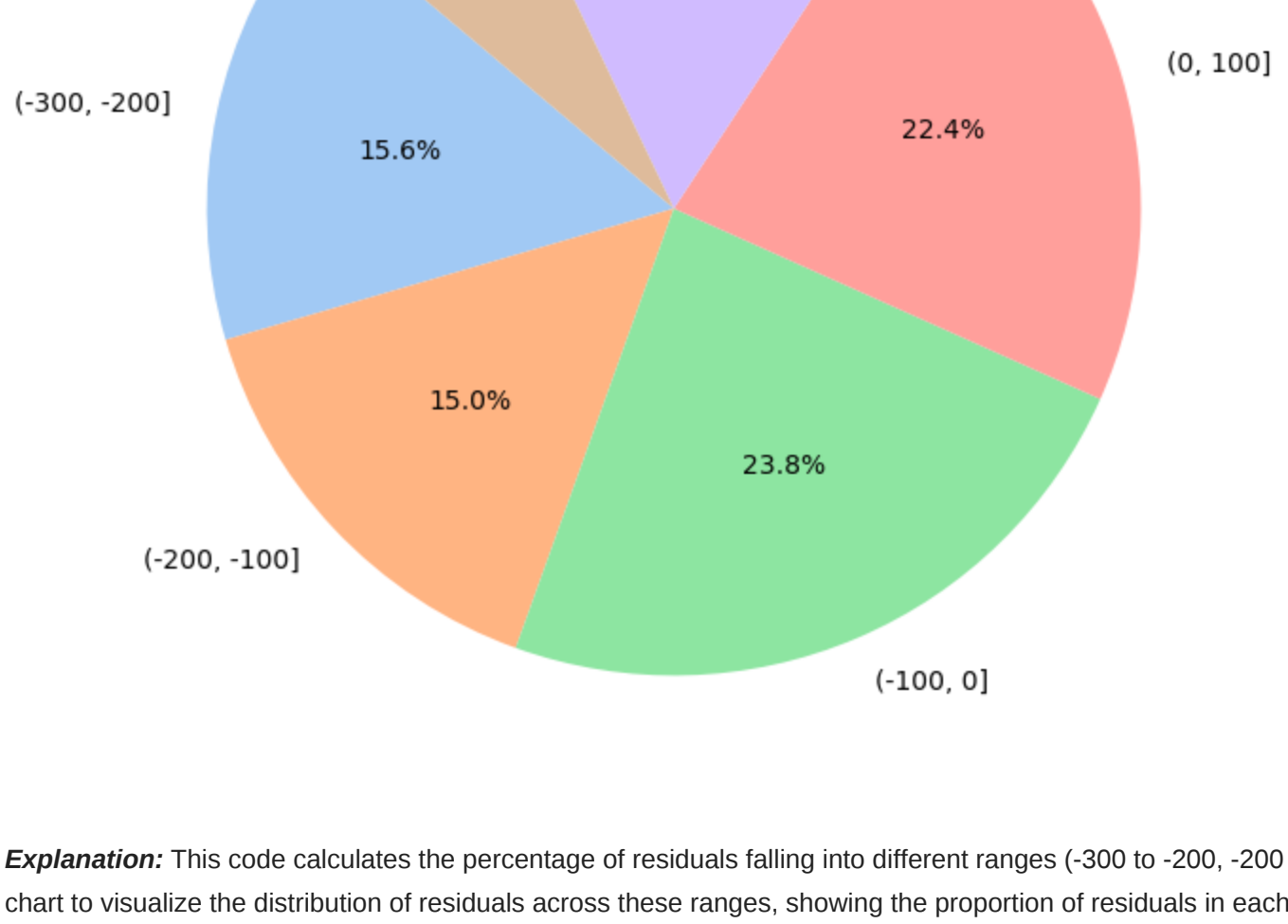
Omnibus:	39.302	**Durbin-Watson:**	1.955
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	82.258
Skew:	1.564	**Prob(JB):**	1.37e-18
Kurtosis:	6.186	**Cond. No.**	6.81e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.81e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Explanation This code first redefines the dataset 'era_950_1050' by selecting rows where the 'expected_recovery_amount' is between 950 and 1050. Then, it defines the independent variable (X) as a combination of 'expected_recovery_amount' and the 'indicator_1000' variable. The dependent variable (y) is set as 'actual_recovery_amount'. After adding a constant term to the independent variables, it builds a linear regression model using Ordinary Least Squares (OLS) and prints the summary statistics of the model. This analysis aims to examine the relationship between the expected recovery amount, the indicator variable, and the actual recovery amount within the specified range.

In [17]:
```python
# Calculate the percentage of residuals in different ranges
residual_ranges = pd.cut(residuals, bins=[-300, -200, -100, 0, 100, 200, 300])
residual_counts = pd.value_counts(residual_ranges, sort=False)
residual_percentages = (residual_counts / len(residuals)) * 100

# Plot pie chart
plt.figure(figsize=(8, 8))
plt.pie(residual_percentages, labels=residual_percentages.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette('pastel'))
plt.title("Pie Chart of Residuals")
plt.show()
```



Explanation: This code calculates the percentage of residuals falling into different ranges (-300 to -200, -200 to -100, -100 to 0, 0 to 100, 100 to 200, 200 to 300). It then creates a pie chart to visualize the distribution of residuals across these ranges, showing the proportion of residuals in each category. The chart is color-coded using a pastel color palette for better visualization. This analysis provides insights into the distribution of errors in the regression model.