Welcome to Colab!

(New) Try the Gemini API

- Generate a Gemini API key
- Talk to Gemini with the Speech-to-Text API
- Gemini API: Quickstart with Python
- Gemini API code sample
- Compare Gemini with ChatGPT
- More notebooks

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view and the command palette.


```
Start coding or generate with AI.
```

What is Colab?

Colab, or 'Colaboratory', allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to find out more, or just get started below!

Getting started

The document that you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut 'Command/Ctrl+Enter'. To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

Files ImportError ×

Please explain the error:

```
ImportError: cannot import name 'RandomUnderSample
```

Hypothesis

The error message `ImportError: cannot import name 'RandomUnderSampler' from 'imblearn.over_sampling'` indicates that you are trying to import `RandomUnderSampler` from the `imblearn.over_sampling` module, but it is not found there. This is because `RandomUnderSampler` is an **under-sampling** technique, and it resides in the `imblearn.under_sampling` module, not the `imblearn.over_sampling` module.

Suggested Changes

```
# Instead of importing from 'imblearn.over_sampl
from imblearn.over_sampling import RandomOverSam


# Now you can create an instance of RandomOverSa
over = RandomOverSampler()
```

Use code with caution

Explanation of changes:

1. **Import from the correct module:** We changed the import statement to `from imblearn.over_sampling import RandomOverSampler` to import the correct class for oversampling. `RandomOverSampler` is used for increasing the number of instances in the minority class.
2. **Use the correct class:** We replaced `RandomUnderSampler` with `RandomOverSampler` in the assignment statement to use the appropriate class for oversampling.

▶ Sources

Rate this answer

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

```
604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To find out more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: Create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To find out more about the Jupyter project, see jupyter.org.

Data science

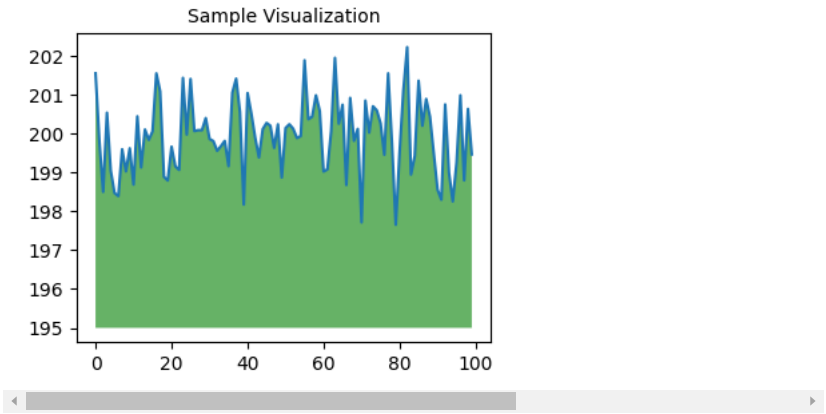
With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualise it. To edit the code, just click the cell and start editing.

```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```



You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from GitHub and many other sources. To find out more about importing data, and how Colab can be used for data science, see the links below under Working with data.

Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

More resources

Working with notebooks in Colab

- Overview of Colaboratory
- Guide to markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets

Working with data

- Loading data: Drive, Sheets and Google Cloud Storage
- Charts: visualising data
- Getting started with BigQuery

Machine learning crash course

These are a few of the notebooks from Google's online machine learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Linear regression with tf.keras using synthetic data

Using accelerated hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

Featured examples

- NeMo voice swap: Use Nvidia NeMo conversational AI toolkit to swap a voice in an audio fragment with a computer-generated one.
- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB film reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine-learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


data=pd.read_excel('/content/SystemAdministrators (2).xlsx')


data.head()
```

	Experience	Training	Completed task
0	10.9	4	Yes
1	9.9	4	Yes
2	10.4	6	Yes
3	13.7	6	Yes
4	9.4	8	Yes

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data.shape
```

```
(75, 3)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75 entries, 0 to 74
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Experience      75 non-null     float64
 1   Training        75 non-null     int64  
 2   Completed task  75 non-null     object 
dtypes: float64(1), int64(1), object(1)
memory usage: 1.9+ KB
```

```
data['Completed task'].value_counts()
```

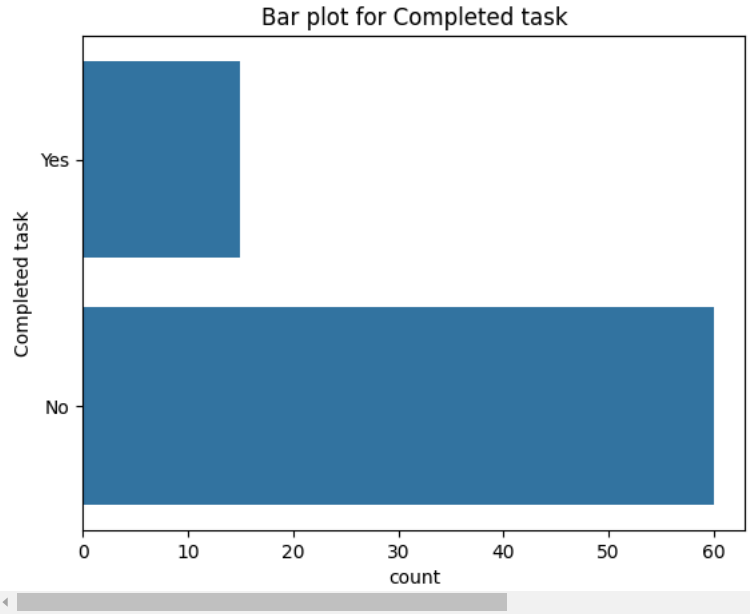
Completed task	count
No	60
Yes	15

dtype: int64

```
#1/3 rde se kam then data is unbalanced otherwise if it is more then data is balance
# to balanced data  .... we use 4 methods A)oversampling to minority B)undersampling
```

```
sns.countplot(data['Completed task'])
plt.title('Bar plot for Completed task')
```

```
Text(0.5, 1.0, 'Bar plot for Completed task')
```




```
#separating  target column from other columns
x=data.drop(columns='Completed task')
y=data['Completed task']
```

x

	Experience	Training
0	10.9	4
1	9.9	4
2	10.4	6
3	13.7	6
4	9.4	8
...	...	...
70	5.6	4
71	5.9	8
72	6.4	6
73	3.8	4
74	5.3	4

75 rows × 2 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

y

	Completed task
0	Yes
1	Yes
2	Yes
3	Yes
4	Yes
...	...
70	No
71	No
72	No
73	No
74	No

75 rows × 1 columns

dtype: object

```
#first method of balancing
#understanding of majority category

from imblearn.under_sampling import RandomUnderSampler

under=RandomUnderSampler()
x_us,y_us=under.fit_resample(x,y)

y_us.value_counts()
```

Completed task	count
No	15
Yes	15

dtype: int64

```
#second method
#oversampling of minority category

from imblearn.over_sampling import RandomOverSampler
```

```
over=RandomOverSampler()
x_os,y_os=over.fit_resample(x,y)
```

```
y_os.value_counts()
```

Completed task	count
Yes	60
No	60

dtype: int64

```
#mathod 3 SMOTE(synthetic minority over sampling technique) #oversampling minority c
```

```
from imblearn.over_sampling import SMOTE
```

```
smote =SMOTE()
x_smote,y_smote=smote.fit_resample(x,y)
```

```
y_smote.value_counts()
```

Completed task	count
Yes	60
No	60

dtype: int64

```
#method 4 SMOOTEEN (COMBINE UNDERSAMPLING AND OVERSAMPLING)
```

```
from imblearn.combine import SMOTEENN
smo=SMOTEENN()
x_smo,y_smo=smo.fit_resample(x,y)
```

```
y_smo.value_counts()
```

Completed task	count
No	46
Yes	39

dtype: int64

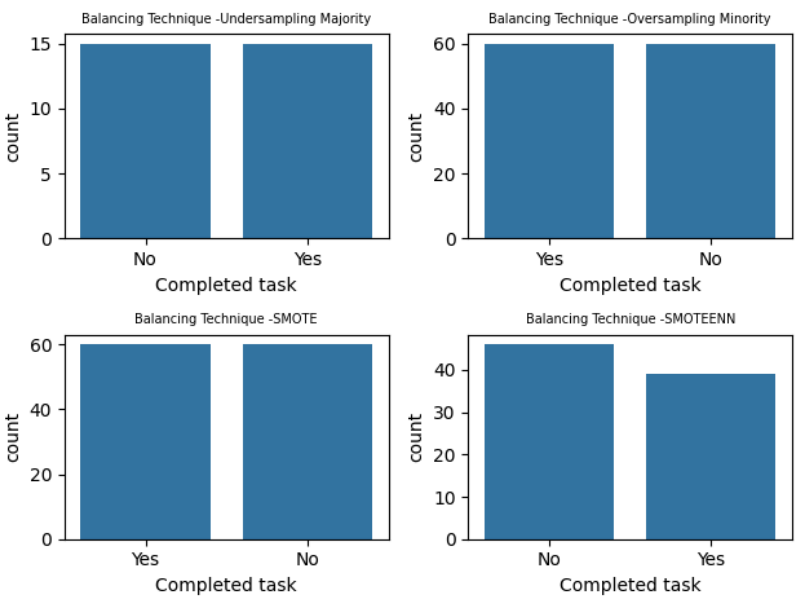
```
#plotting bar plots for all 4 methods
```

```
plt.subplot(2,2,1)
plt.title('Balancing Technique -Undersampling Majority',fontsize=7)
sns.countplot(x=y_us)

plt.subplot(2,2,2)
plt.title('Balancing Technique -Oversampling Minority',fontsize=7)
sns.countplot(x=y_os)

plt.subplot(2,2,3)
plt.title('Balancing Technique -SMOTE',fontsize=7)
sns.countplot(x=y_smote)

plt.subplot(2,2,4)
plt.title('Balancing Technique -SMOTEENN',fontsize=7)
sns.countplot(x=y_smo)
plt.tight_layout()
```



nclusion => we can conclude from this from this 4 method 1 method is not giving equal

Start coding or generate with AI.

Enter a prompt here

0/2000

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more