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

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:

```
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

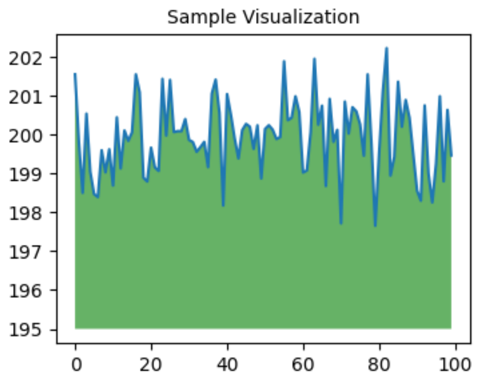
With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualise it. To edit the code, just click the cell and start editing.

```
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"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```



Start coding or generate with AI.

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.

```
#Polynomial Regrssion Modelling
```

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

```
df=pd.read_csv("/content/IcecreameSale.csv")
df.head()
```

	Temperature	Sales
0	-4.662263	41.842986
1	-4.316559	34.661120
2	-4.213985	39.383001
3	-3.949661	37.539845
4	-3.578554	32.284531

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

```
df.shape
```

```
(49, 2)
```

```
df.info
```

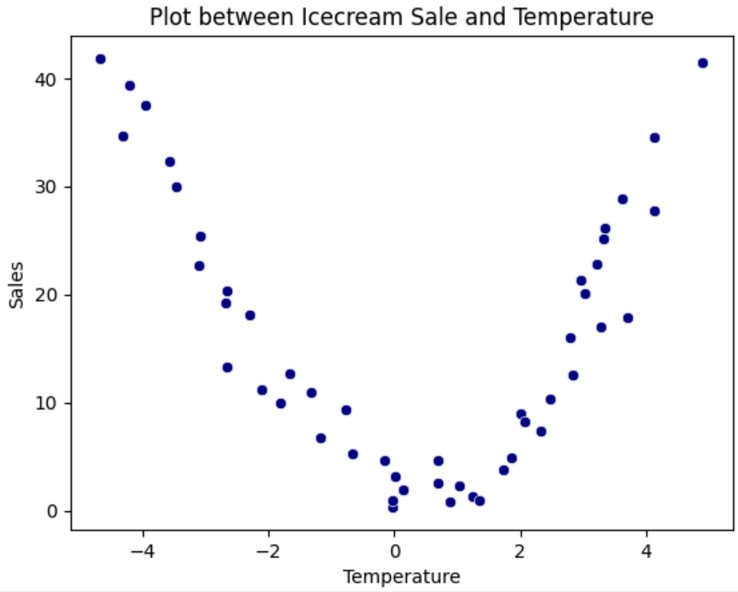
```
pandas.core.frame.DataFrame.info
def info(verbose: bool | None=None, buf: WriteBuffer[str] | None=None, max_cols: int | None=None,
memory_usage: bool | str | None=None, show_counts: bool | None=None) -> None
```

```
/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py
Print a concise summary of a DataFrame.

This method prints information about a DataFrame including
the index dtype and columns, non-null values and memory usage.
```

```
#plotting original data
sns.scatterplot(x=df['Temperature'],y=df['Sales'],color='navy')
plt.title('Plot between Icecream Sale and Temperature')
```

```
Text(0.5, 1.0, 'Plot between Icecream Sale and Temperature')
```

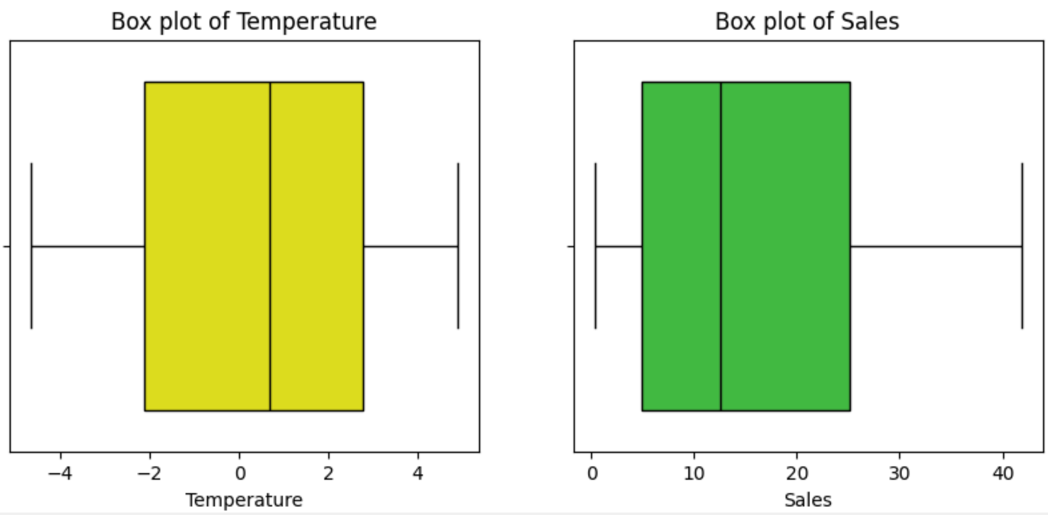



```
#we observe curve in the graph which suggestes nonlinear relation
#between x and y
#therefore linear regression will not work
#we must use of polynomilal regression to fit the data


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
sns.boxplot(df['Temperature'],color='yellow',fill=True,orient='h',linecolor='black')
plt.title('Box plot of Temperature')
plt.subplot(1,2,2)
sns.boxplot(df['Sales'],color='limegreen',fill=True,orient='h',linecolor='black')
plt.title('Box plot of Sales')
```

```
Text(0.5, 1.0, 'Box plot of Sales')
```




```
#scaling of columns
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
scaler.fit(df)
```

```
▾ StandardScaler
StandardScaler()
```

```
cols=df.columns
scaled_df=scaler.transform(df)
new_df=pd.DataFrame(scaled_df,columns=cols)
new_df.head()
```

	Temperature	Sales
0	-1.847945	2.136743
1	-1.718468	1.545101
2	-1.680051	1.934089
3	-1.581053	1.782250
4	-1.442062	1.349318

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

```
#checking correlation between temperatur and Sales
correlation=new_df.corr()['Sales'].sort_values(ascending=True)
correlation
```

	Sales
Temperature	-0.175184
Sales	1.000000

dtype: float64

Generated code may be subject to a licence | Korjick/ML

```
#Splitting dataset for regression Modelling
x=new_df[['Temperature']]#making series as dataframe
y=new_df['Sales']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

```
x_train.shape
```

```
(39, 1)
```

```
y_train.shape
```

```
(39,)
```

```
x_test.shape
```

```
(10, 1)
```

```
y_test.shape
```

```
(10,)
```

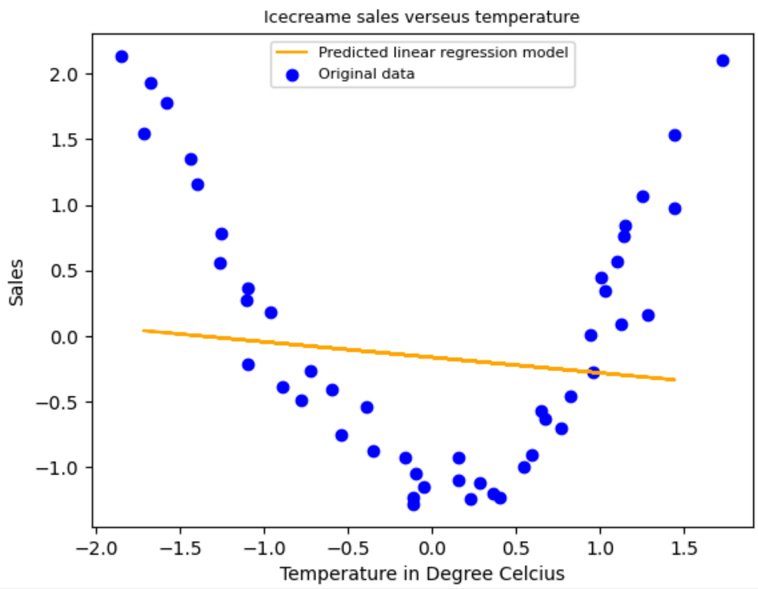
```
#Applying first linear regression
from sklearn.linear_model import LinearRegression
```

```
model1=LinearRegression()
model1.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
from sklearn.metrics import mean_squared_error,r2_score
```

```
plt.plot(x_train,model1.predict(x_train),color='orange',label='Predicted linear regression model')
plt.scatter(x,y,color='blue',label='Original data')
plt.title('Linear Regression Model')
plt.xlabel('Temperature in Degree Celcius')
plt.ylabel('Sales')
plt.title('Icecreame sales verseus temperature',fontsize=9)
plt.legend(loc='upper center',fontsize=8)
plt.show()
```




```
#calculate r2 for training and testing
y_train_pred=model1.predict(x_train)
y_test_pred=model1.predict(x_test)

r2_train=r2_score(y_train,y_train_pred)
r2_test=r2_score(y_test,y_test_pred)

print(f"Training R2 score={r2_train: .2f}")
print(f"Testing R2 score={r2_test:.2f}")
```

```
Training R2 score= 0.02
Testing R2 score=-0.41
```

```
#Training and testing is poor here
#It shows that training and testing R2 score are very less
#therfore simple linear regression is not fit
```

```
#Fitting  Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
```

```
poly=PolynomialFeatures(degree=2,include_bias=False)
```

```
#Aplying polynomial fitting to train and test data

x_train_poly=poly.fit_transform(x_train)
x_test_poly=poly.fit_transform(x_test)
# The target variable (y) should not be transformed. Remove the following lines:
#y_train_poly=poly.fit_transform(y_train)
#y_test_poly=poly.fit_transform(y_test)
```

```
model2=LinearRegression()
model2.fit(x_train_poly,y_train)
```

```
▾ LinearRegression ⓘ ?
LinearRegression()
```

```
y_train_pred_poly=model2.predict(x_train_poly)
y_test_pred_poly=model2.predict(x_test_poly)
```

```
r2_train_poly=r2_score(y_train,y_train_pred_poly)
r2_test_poly=r2_score(y_test,y_test_pred_poly)

print(f"Training R2 score={r2_train_poly: .2f}")
print(f"Testing R2 score={r2_test_poly:.2f}")
```

```
Training R2 score= 0.92
Testing R2 score=0.92
```

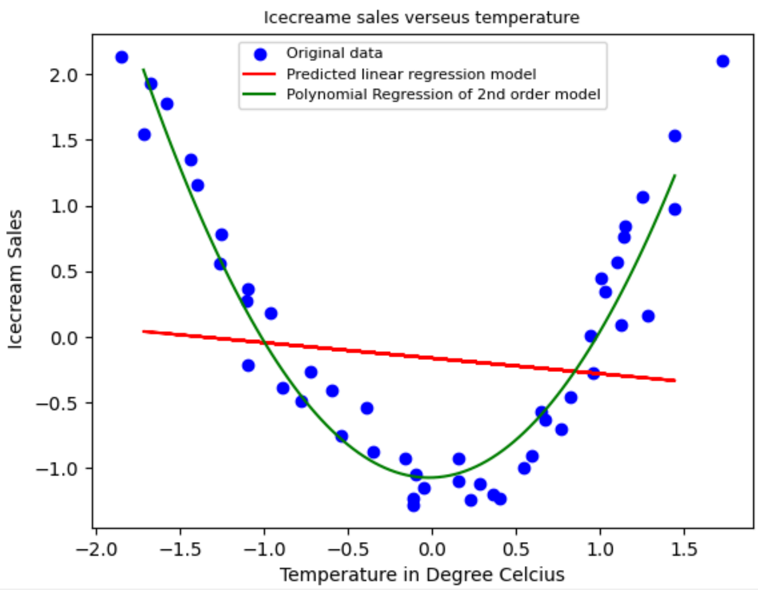
```
#we observe that R2 score is improved in polynomial regression compared to linear regression
```

```
#plotting both regression with original data for comparison

myline =np.linspace(x_train.min(),x_train.max(),100)
mymodel=np.poly1d(np.polyfit(new_df['Temperature'],new_df['Sales'],2))

plt.scatter(x,y,color='blue',label='Original data')
plt.plot(x_train,model1.predict(x_train),color='red',label='Predicted linear regression model')
plt.plot(myline,mymodel(myline),color='green',label='Polynomial Regression of 2nd order model')

plt.title('Comparison of linear and polynomial regression')
plt.xlabel('Temperature in Degree Celcius')
plt.ylabel(' Icecream Sales')
plt.title('Icecreame sales verseus temperature',fontsize=9)
plt.legend(loc='upper center',fontsize=8)
plt.show()
```




```
#polynomial regression best fits the given data compares to linear regression
```

Start coding or generate with AI.