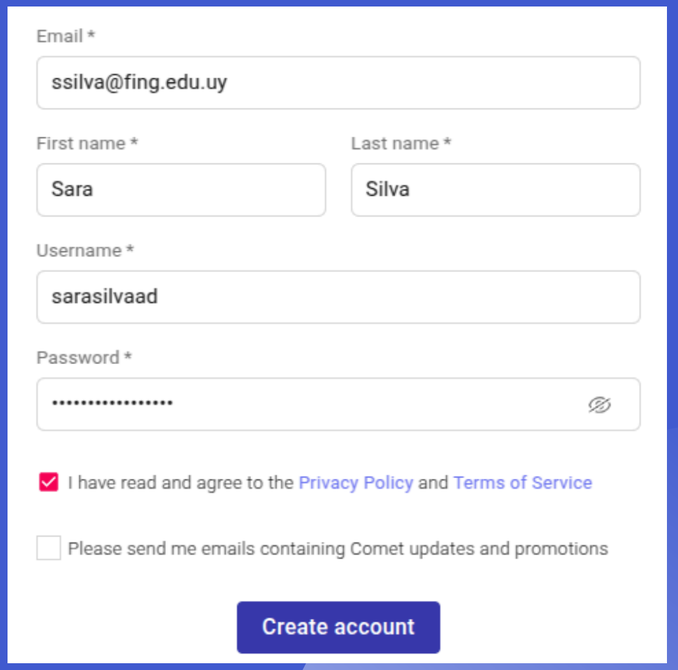
comet

IDEA

Comet es una plataforma que ayuda a hacer **seguimiento**, **visualización** y **comparación** de experimentos de machine learning. Permite registrar métricas, hiperparámetros, gráficos y modelos en tiempo real.

CREAR USUARIO

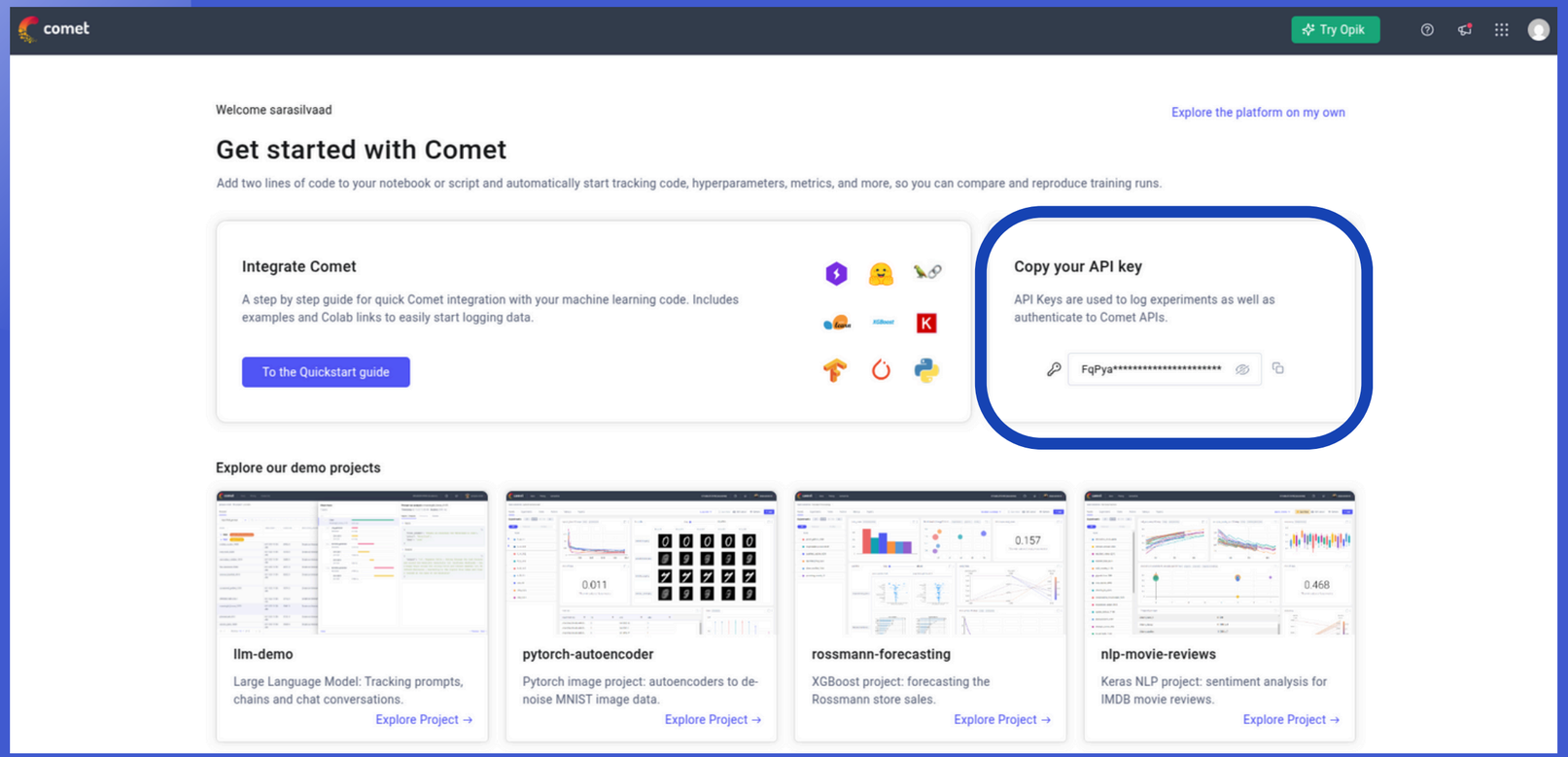
https://www.comet.com/login

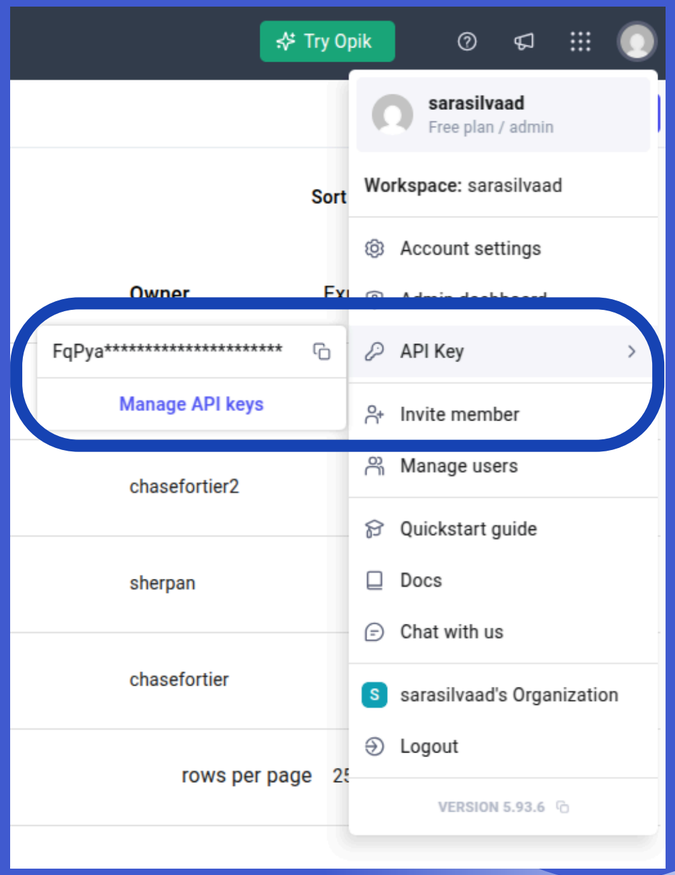


API KEY

identificador único que los autentica ante un servicio.

Esta clave le dice a Comet quiénes son ustedes y les permite conectarse a sus servicios desde su código, sin necesidad de ingresar usuario y contraseña cada vez.





EN EL CÓDIGO

```
!pip install comet_ml
```

Mostrar el resultado oculto

```
[5] #import comet_ml in the top of your file
    from comet_ml import Experiment
```

```
[8] API_KEY='FqPy[redacted]'
```

Instanciamos experimento

```
#create an experiment with your api key
exp = Experiment(api_key=API_KEY,  #api_key=os.environ.get("COMET_API_KEY"),
                 project_name='movie_reviews',
                 auto_param_logging=False)
exp.set_name('parte4')
exp.add_tags(['basico'])
```

Si el proyecto no existe ya, lo va a crear

Creamos un experimento y le indicamos a qué proyecto pertenece, el nombre del experimento, y algún tag que querramos ponerle

- Entrenamos -> podría ser este como cualquier otro entrenamiento

```
[31] basic_pipeline_scores = cross_validate(basic_pipeline, X4, y4, scoring=['accuracy', 'precision', 'recall'])
```

Loggeamos los resultados

```
exp.log_dataset_hash(X4)
exp.log_parameters(basic_pipeline.get_params())
save_results(exp, basic_pipeline_scores)
exp.end()
```

```
# X4 = X_train.values
```

Nativo de la clase Experiment de Comet

Función que definimos nosotros en función del formato de nuestros resultados y lo que quiero reportar

Ejemplo de función para reportar resultados:

```python
def save_results(cmt_exp, cross_val_results):
    '''
    Entrada:
        cmt_exp: experimento comet
        cross_val_scores: scikit-learn cross validation results
    '''
    avg_fit_time = cross_val_results['fit_time'].mean()
    std_fit_time = cross_val_results['fit_time'].std()
    avg_cv_accuracy = cross_val_results['test_accuracy'].mean()
    std_cv_accuracy = cross_val_results['test_accuracy'].std()
    avg_cv_precision = cross_val_results['test_precision'].mean()
    std_cv_precision = cross_val_results['test_precision'].std()
    avg_cv_recall = cross_val_results['test_recall'].mean()
    std_cv_recall = cross_val_results['test_recall'].std()


    metrics = {
        'avg_fit_time':avg_fit_time,
        'std_fit_time':std_fit_time,
        'avg_cv_accuracy':avg_cv_accuracy,
        'std_cv_accuracy':std_cv_accuracy,
        'avg_cv_precision':avg_cv_precision,
        'std_cv_precision':std_cv_precision,
        'avg_cv_recall':avg_cv_recall,
        'std_cv_recall':std_cv_recall,
    }

    plt.figure()
    plt.plot(cross_val_results['test_accuracy'],'*-')
    plt.xlabel('fold')
    plt.ylabel('accuracy')
    plt.title('Cross validation accuracy')
    plt.grid()


    exp.log_metrics(metrics)
    exp.log_figure(figure=plt)

    return
```

Escribimos un diccionario con un valor por métrica

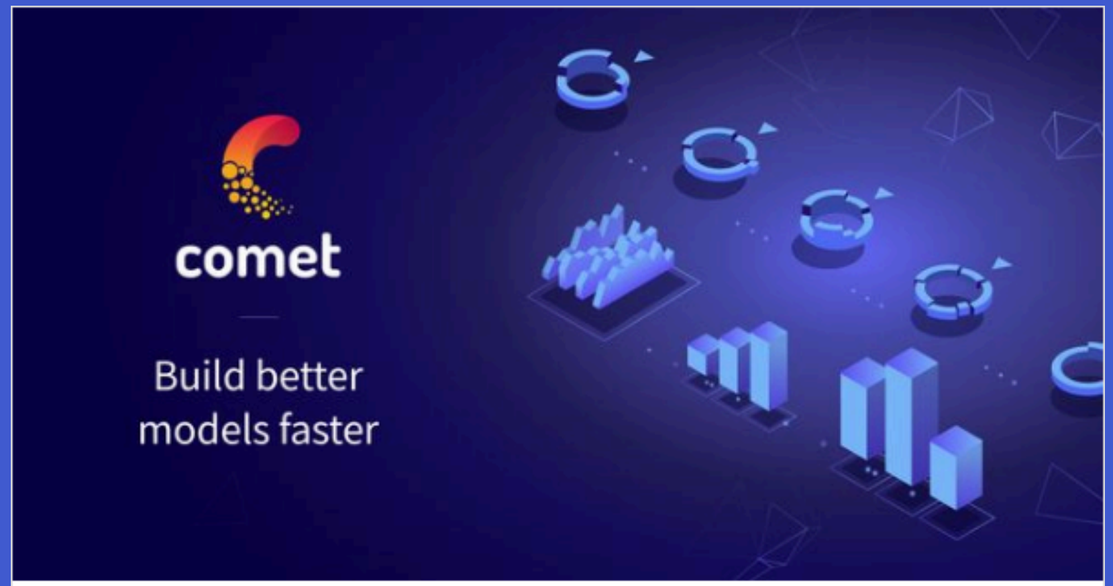
Hacemos un plot

Loggeamos tanto las métricas como la figura

```python
basic_pipeline_scores
```

```
{'fit_time': array([45.23854232, 44.71766424, 46.65585852, 41.33927178, 46.95525622]),
 'score_time': array([1.31294918, 1.21357536, 1.17131662, 1.17548633, 1.18535161]),
 'test_accuracy': array([0.88185714, 0.88585714, 0.88314286, 0.88014286, 0.88342857]),
 'test_precision': array([0.8799431 , 0.88392603, 0.87746479, 0.87953183, 0.87881356]),
 'test_recall': array([0.88421955, 0.88822184, 0.89050886, 0.88078902, 0.88936535])}
```

Y ESTO CÓMO SE VE EN COMET?



Supercharging Machine Learning

Comet lets you track code, experiments, and results on ML projects. It’s fast, simple, and free for open source projects.

Comet.ml

DOCUMENTACIÓN DE EXPERIMENTOS

https://www.comet.com/docs/python-sdk/Experiment/