# **Prompt engineering documentation**

## La première tentatative, on a demandé à chat gpt

# La question que vous voyez ci-dessus :

Aide moi à modifier ce code pour pouvoir utilisé les modèles suivants:XGBoost Classifier • LightGBM Classifier • CatBoost Classifier from sklearn.naive\_bayes import GaussianNB # Initialize the Naive Bayes model naive\_bayes\_model = GaussianNB() # Fit the model to the training data naive\_bayes\_model.fit(X\_train, y\_train) # Make predictions on the test data y\_pred\_naive\_bayes = naive\_bayes\_model.predict(X\_test) # Calculate probabilities for ROC-AUC score y\_prob\_naive\_bayes = naive\_bayes\_model.predict\_proba(X\_test) # Update the metrics dictionary metrics\_dict['Model'].append('Naive Bayes')  $metrics\_dict['Accuracy'].append(accuracy\_score(y\_test,$ y\_pred\_naive\_bayes)) metrics\_dict['Precision'].append(precision\_score(y\_test, y\_pred\_naive\_bayes, average='weighted'))  $metrics\_dict['Recall'].append(recall\_score(y\_test,$ y\_pred\_naive\_bayes, average='weighted'))  $metrics\_dict['F1-Score'].append(f1\_score(y\_test,$ v nred naive haves average='weighted')) metrics\_dict['AUC-ROC'].append(roc\_auc\_score(y\_test, y\_prob\_naive\_bayes, multi\_class='ovr', average='weighted')) # Update the DataFrame to display the results metrics\_df = pd.DataFrame(metrics\_dict) # Sort the DataFrame by AUC-ROC in descending order metrics\_df\_sorted = metrics\_df.sort\_values(by='AUC-ROC', ascending=False) metrics\_df\_sorted

Il nous a sorti ce code ci- dessous qui est compliqué difficile à Comprendre.

```
from xgboost import XGBClassifier

	☐ Copier

from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
# Initialize the models
xgboost_model = XGBClassifier(eval_metric='mlogloss', use_label_encoder=False)
lightgbm model = LGBMClassifier()
catboost model = CatBoostClassifier(silent=True)
# Fit the models to the training data
xgboost_model.fit(X_train, y_train)
lightgbm_model.fit(X_train, y_train)
{\tt catboost\_model.fit}({\tt X\_train,\ y\_train})
# Make predictions on the test date
y_pred_xgboost = xgboost_model.predict(X_test)
y pred lightgbm = lightgbm model.predict(X test)
y_pred_catboost = catboost_model.predict(X_test)
# Calculate probabilities for ROC-AUC scor
y_prob_xgboost = xgboost_model.predict_proba(X_test)
y_prob_lightgbm = lightgbm_model.predict_proba(X_test)
y_prob_catboost = catboost_model.predict_proba(X_test)
# Update the metrics dictionary for each model
metrics_dict['Model'].extend(['XGBoost', 'LightGBM', 'CatBoost'])
metrics_dict['Accuracy'].extend([
     accuracy_score(y_test, y_pred_xgboost),
     accuracy_score(y_test, y_pred_lightgbm),
     accuracy_score(y_test, y_pred_catboost)
metrics_dict['Precision'].extend([
    precision_score(y_test, y_pred_xgboost, average='weighted'),
     precision_score(y_test, y_pred_lightgbm, average='weighted'),
    precision_score(y_test, y_pred_catboost, average='weighted')
metrics_dict['Recall'].extend([
    recall_score(y_test, y_pred_xgboost, average='weighted'),
     recall_score(y_test, y_pred_lightgbm, average='weighted'),
     recall_score(y_test, y_pred_catboost, average='weighted')
metrics_dict['F1-Score'].extend([
                                                                Ø Copier %
    f1_score(y_test, y_pred_xgboost, average='weighted'),
    f1_score(y_test, y_pred_lightgbm, average='weighted'),
    f1_score(y_test, y_pred_catboost, average='weighted')
metrics_dict['AUC-ROC'].extend([
    roc_auc_score(y_test, y_prob_xgboost, multi_class='ovr', average='weighted'),
    roc_auc_score(y_test, y_prob_lightgbm, multi_class='ovr', average='weighted'),
    roc_auc_score(y_test, y_prob_catboost, multi_class='ovr', average='weighted')
metrics_df = pd.DataFrame(metrics_dict)
                                                                      ⊕ Cop
# Update the DataFrame to display the results
metrics_df = pd.DataFrame(metrics_dict)
# Sort the DataFrame by AUC-ROC in descending order
metrics_df_sorted = metrics_df.sort_values(by='AUC-ROC', ascending=False)
 # Display the sorted DataFrame
metrics_df_sorted
```

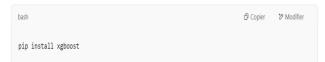
# La deuxième tentative qui a donné un code explicite et clair :

il faut séparer le code pour chaque model c'est à dire 1 à 1

D'accord, je vais détailler les modifications pour chaque modèle l'un après l'autre.

## 1. XGBoost Classifier

Installation de XGBoost : Avant de l'utiliser, assure-toi que XGBoost est installé :



## Code modifié pour XGBoost :



#### 2. LightGBM Classifier

Installation de LightGBM : Avant de l'utiliser, assure-toi que LightGBM est installé :

bash	<b>□</b> Copier	
pip install lightgbm		

### Code modifié pour LightGBM:



Et c'est ce code qui nous a aidé à entrainer nos models