Prompt engineering documentation

La première tentatative, on a demandé à chat gpt

La question que vous voyez ci-dessus :

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

```
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, y_pred_naive_bayes, 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
```

```
from xgboost import XGBClassifier
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)
catboost_model.fit(X_train, y_train)

# Make predictions on the test data
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 score
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([
    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')
])

# Update the DataFrame to display the results
metrics_df = pd.DataFrame(metrics_dict)

# 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 :

bash — Copier — Modifier

```
pip install xgboost
```

Code modifié pour XGBoost :

```
from xgboost import XGBClassifier

# Initialisation du modèle XGBoost
xgboost_model = XGBClassifier(eval_metric='mlogloss', use_label_encoder=False)

# Entraînement du modèle
xgboost_model.fit(X_train, y_train)

# Prédictions sur les données de test
y_pred_xgboost = xgboost_model.predict(X_test)

# Probabilités pour le calcul de l'AUC-ROC
y_prob_xgboost = xgboost_model.predict_proba(X_test)
```

Copier — Modifier

```
# Mise à jour des métriques pour XGBoost
metrics_dict['Model'].append('XGBoost')
metrics_dict['Accuracy'].append(accuracy_score(y_test, y_pred_xgboost))
metrics_dict['Precision'].append(precision_score(y_test, y_pred_xgboost, average='weighted'))
metrics_dict['Recall'].append(recall_score(y_test, y_pred_xgboost, average='weighted'))
metrics_dict['F1-Score'].append(f1_score(y_test, y_pred_xgboost, average='weighted'))
metrics_dict['AUC-ROC'].append(roc_auc_score(y_test, y_prob_xgboost, multi_class='ovr', averag

# Affichage des résultats
metrics_df = pd.DataFrame(metrics_dict)
metrics_df_sorted = metrics_df.sort_values(by='AUC-ROC', ascending=False)
metrics_df_sorted
```

2. LightGBM Classifier

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

bash — Copier — Modifier

```
pip install lightgbm
```

Code modifié pour LightGBM :

python — Copier — Modifier

```
from lightgbm import LGBMClassifier

# Initialisation du modèle LightGBM
lightgbm_model = LGBMClassifier()

# Entraînement du modèle
lightgbm_model.fit(X_train, y_train)

# Prédictions sur les données de test
y_pred_lightgbm = lightgbm_model.predict(X_test)

# Probabilités pour le calcul de l'AUC-ROC
y_prob_lightgbm = lightgbm_model.predict_proba(X_test)

# Mise à jour des métriques pour LightGBM
metrics_dict['Model'].append('LightGBM')
```

Copier — Modifier

```
# Initialisation du modèle LightGBM
lightgbm_model = LGBMClassifier()

# Entraînement du modèle
lightgbm_model.fit(X_train, y_train)

# Prédictions sur les données de test
y_pred_lightgbm = lightgbm_model.predict(X_test)

# Probabilités pour le calcul de l'AUC-ROC
y_prob_lightgbm = lightgbm_model.predict_proba(X_test)

# Mise à jour des métriques pour LightGBM
metrics_dict['Model'].append('LightGBM')
metrics_dict['Accuracy'].append(accuracy_score(y_test, y_pred_lightgbm))
metrics_dict['Precision'].append(precision_score(y_test, y_pred_lightgbm, average='weighted'))
metrics_dict['Recall'].append(recall_score(y_test, y_pred_lightgbm, average='weighted'))
metrics_dict['F1-Score'].append(f1_score(y_test, y_pred_lightgbm, average='weighted'))
metrics_dict['AUC-ROC'].append(roc_auc_score(y_test, y_prob_lightgbm, multi_class='ovr', averag

# Affichage des résultats
metrics_df = pd.DataFrame(metrics_dict)
metrics_df_sorted = metrics_df.sort_values(by='AUC-ROC', ascending=False)
metrics_df_sorted
```

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