GitHub Link- https://github.com/rakshitgarg99/Rapids_data_science/tree/main/Lab_Evaluation • Kaggle Link- https://www.kaggle.com/rakshitgarg99/netflix-classification Libraries used- cuPy, cuDF, cuML, sklearn, pandas, matplotlib, seaborn Models Implemented- Logistic Regression, Random Forest, Xgboost, LGBM Evaluation Metrics Used- AUC Kaggle Rank Achieved with total number of teams- 40th Rank out of 147 (On-going contest) Tasks done in code-1. Data Preprocessing (EDA, Handling null values, High cardinality features, Numerical and categorical features, Temporal or datetime features and coorelation) 2. Model building and evaluation(AUC) 3. Detailed Coding documentationm Proof of Rank-40 **Rakshit Garg** 0.78698 6 4h Your Best Entry! Your submission scored 0.78544, which is not an improvement of your previous score. Keep trying! Netflix Appetency - Classify consumers according to their appetite to subscribe to Netflix. More information about this contest:-1. The metric used is AUC 2. The competition will last 2 months until 2022-03-31 3. The test file is split into 2 parts, 20% for the public classification and 80% for the private classification. Important - As it's an ongoing contest and only 20% of data is used in the evaluation. We cannot implement a more optimized model because of the fear of overfitting. I am attaching some threads of the discussion section; please go through them. https://www.kaggle.com/c/netflix-appetency/discussion/303029 https://www.kaggle.com/c/netflix-appetency/discussion/304284 With the above information from the host and discussion section, I have done the assignment considering the best score will be 0.7878 with a variance of +/-0.0024 # This Python 3 environment comes with many helpful analytics libraries installed In [94]: # It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python # For example, here's several helpful packages to load import numpy as np # linear algebra import pandas as pd # data processing, CSV file I/O (e.g. pd.read csv) # Input data files are available in the read-only "../input/" directory # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input direc import os for dirname, , filenames in os.walk('/kaggle/input'): for filename in filenames: print(os.path.join(dirname, filename)) # You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you # You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session Importing libraries In [95]: import cupy as np import cudf as pd import matplotlib.pylab as plt import seaborn as sns **Getting Input Files** train_df = pd.read_csv("../input/netflix-appetency/train.csv") test_df = pd.read_csv("../input/netflix-appetency/test.csv") ss_df = pd.read_csv("../input/netflix-appetency/sample_submission.csv") train df.shape, test df.shape, ss df.shape EDA - Expolratory Data Analysis sns.countplot(train df.target.to array()) plt.show() train df.head() We can observe that lots for featues have NaN values train df.isna().sum(axis=0).sort values(ascending=False).to frame().T we can see that some features have all null values, we can drop those null columns = ['feature 193', 'feature 196', 'feature 197', 'feature 198', 'feature 83'] train df = train df.drop(columns=null columns) test df = test df.drop(columns=null columns) train df.shape, test df.shape train df.info() Categorical Colummns train_df.select_dtypes('object').head() In [104... we can observe that those are catgorical features encoded with some labels • we can also observe that some date features, we can drop those as of now In [107... ## getting date columns cat_columns = [col for col in train_df.select_dtypes('object').columns] $mask = train_df[cat_columns][:10].to_pandas().astype(str).apply(lambda x : x.str.match('(\d{2,4}(-|\/|\\|.|)))) | (ambda x : x.str.match('(\d{2,4}(-|\/|\\|.|)))) | (base x : x.str.match('(\d{2,4}(-|\/|\\|.|)))) | (base x : x.str.match('(\d{2,4}(-|\/|\\|.|)))) | (base x : x.str.match('(\d{2,4}(-|\/|\\|.|)))) | (catalog x : x.str.match('(\d{2,4}(-|\/|\|.|)))) | (catalog x : x.str.match('(\d{2,4}(-|\/|\|$ In [134... mask idx = list(mask.index) for i in range(len(mask)): if mask[i]==True: print(mask idx[i]) In [74]: date_columns = ['feature_191', 'feature_192', 'feature_194', 'feature_195', 'feature_199', 'feature_200', 'feature_196', 'feature_196', 'feature_196', 'feature_196', 'feature_196', 'feature_196', 'feature_200', 'feat display(train_df[date_columns].head()) train df = train df.drop(columns=date columns) test_df = test_df.drop(columns=date_columns) train df.shape, test df.shape # Number of unique categories train_df.to_pandas().select_dtypes('object').nunique().sort_values().to_frame().T we can see that some featues have only single categorie, so we can drop those featues single_feature_columns = ['feature_190', 'feature_55', 'feature_187', 'feature_188', 'feature_249', 'feature_24 train df = train df.drop(columns=single feature columns) test_df = test_df.drop(columns=single_feature_columns) train_df.shape, test_df.shape cat columns = [col for col in train df.select dtypes('object').columns] train df[cat columns].head() **Numerical Columns** train_df.select_dtypes(['int', 'float']).head() we can observe that nan values also present in numarical columns target column = 'target' drop_columns = ['id', target_column] num columns = [col for col in train df.select dtypes(['int', 'float']).columns if col not in drop columns] train df[num columns].head() # Number of unique values train df[num columns].to pandas().nunique().sort values().to frame().T #@.columns[30:45] single feature columns = [In [24]: 'feature 461', 'feature 474', 'feature 472', 'feature 409', 'feature 469', 'feature 467', 'feature 411', 'feature 465', 'feature_312', 'feature_463', 'feature_313', 'feature_375', 'feature_427', 'feature_412', 'feature_413', 'feature_455', 'feature_453', 'feature_419', 'feature_451', 'feature_100', 'feature 449', 'feature 421', 'feature 445', 'feature 433', 'feature_432', 'feature_459', 'feature_476', 'feature_405', 'feature 425', 'feature 498', 'feature 227', 'feature 228', 'feature_149', 'feature_495', 'feature_385', 'feature_493', 'feature 478', 'feature 491', 'feature 389', 'feature 489', 'feature_252', 'feature_487', 'feature_502', 'feature_505', 'feature_484', 'feature_395', 'feature_482', 'feature 377', 'feature 358', 'feature 480', 'feature 397', 'feature 357', 'feature_500'] train df = train df.drop(columns=single feature columns) test df = test df.drop(columns=single feature columns) train_df.shape, test_df.shape num columns = [col for col in train df.select dtypes(['int', 'float']).columns if col not in drop columns] In [84]: Handling NAN values using median for numerical features whereas mode for categorical features for col in num columns: train df[col].fillna(train df[col].median(),inplace=True) test df[col].fillna(test df[col].median(),inplace=True) def get mode(feature, df): return df[feature].mode()[0] for col in cat columns: train df[col].fillna(get mode(col,train df),inplace=True) test df[col].fillna(get mode(col, test df),inplace=True) checking correlation of numerical features and target corr_df = train_df[['target'] + num_columns].corr() corr df.head() Label Encoding - encode the categorical features from cuml.preprocessing.LabelEncoder import LabelEncoder # label encoding those cat columns def label encoding(train df, test df, columns): for col in columns: le = LabelEncoder() values = train_df[col].append(test_df[col]) le.fit(values) train_df[col] = le.transform(train_df[col]) test df[col] = le.transform(test df[col]) return train_df, test_df train df, test df = label_encoding(train_df, test_df, cat_columns) train_df[cat_columns].head() target col = 'target' train cols = cat columns + num columns Modeling Logistic Regression from cuml.linear_model import LogisticRegression In [92]: reg = LogisticRegression() reg.fit(train df[train cols].astype('float32'),train df[target col].astype('float32')) y pred = reg.predict proba(test df[train cols].astype('float32')) ss df['target'] = y pred[1] ss df.to csv("submission 1 logReg.csv", index=False) ss df.head() Score of Logistic Regression Leaderboard C Refresh ★ Raw Data YOUR RECENT SUBMISSION Score: 0.50027 submission_1_logReg.csv Submitted by Rakshit_tiet · Submitted just now Private score: ↓ Jump to your leaderboard position Random Forest Classifier from cuml.ensemble import RandomForestClassifier as cuRFC In [43]: cuml model = cuRFC(n bins=156, n_estimators=1450,random_state=42, max_depth=5, split_criterion='gini') cuml_model.fit(train_df[train_cols].astype('float32'),train df[target col].astype('float32')) y predict = cuml model.predict proba(test df[train cols].astype('float32')) y predict[1] ss df['target'] = y_predict[1] ss df.to csv("submission 2 rfc.csv", index=False) ss df.head() Score of Random Forest Classifier Leaderboard C Refresh YOUR RECENT SUBMISSION Score: 0.75139 submission_2_rfc.csv Submitted by Rakshit_tiet · Submitted just now Private score: Jump to your leaderboard position XGB Below is the implementation for cuML based XGBClassifer. But I am getting a error while submitting, Moreover the cuML based XGB didn't make us used the "n_esimators" parameters. I tried to find a solution on https://docs.rapids.ai/api/cuml/stable/api.html#forest-inferencing . But didn't get any suitable way to solve problem. Attaching screenshots regarding the same. n estimators not used [12:13:57] WARNING: ../src/learner.cc:576: Parameters: { "n_estimators" } might not be used. This could be a false alarm, with some parameters getting used by language bindings but then being mistakenly passed down to XGBoost core, or some parameter actually being used but getting flagged wrongly here. Please open an issue if you find any such cases. Failure during submission Leaderboard C Refresh YOUR RECENT SUBMISSION submission_4_xgb.csv **FAILED** Submitted by Rakshit Garg · Submitted 27 minutes ago Submission Error ERROR: Could not parse '' into expected type of Double (Line 14002, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14003, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14004, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14005, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14006, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14007, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14008, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14009, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14010, Column 7) ERROR: Could not parse '' into expected type of Double (Line 14011, Column 7) In [28]: # # from xgboost import XGBClassifier # import xgboost as xgb # from cuml import ForestInference # from cuml.test.utils import array equal # from cuml.model selection import train test split # def train xgboost model(X train, y train, num rounds, model path): # set the xgboost model parameters params = { 'n_estimators': 14140, 'learning rate': 0.011, 'max depth': 4, 'reg alpha': 2.97, 'reg lambda': 2.214, 'scale pos weight': 0.9, 'subsample': 0.776, 'gamma': 0.5} dtrain = xgb.DMatrix(X train, label=y train) # train the xgboost model bst = xgb.train(params, dtrain, num rounds) # save the trained xgboost model bst.save model(model path) return bst # def predict xgboost model(X validation, y validation, xgb model): # predict using the xgboost model dvalidation = xgb.DMatrix(X validation, label=y validation) xgb preds = xgb model.predict(dvalidation) # convert the predicted values from xgboost into class labels xgb preds = np.around(xgb preds) return xgb preds # model path = 'xgb.model' # # num of iterations for which the model is trained # num rounds = 15 # # xgboost_model = train_xgboost_model(X_train, y_train, num rounds, model path) # # fil_predsxgb = predict_xgboost_model(X_valid,y_valid,xgboost_model) # trained_model_preds = predict_xgboost_model(X_valid, y valid, xgboost model) # fm = ForestInference.load(filename=model path, algo='BATCH TREE REORG', output class=True, threshold=0.50, model type='xgboost') # test predictions = fm.predict(X valid) Because of the above problems I decided to implement the xgb and lgbm in sklearn to score a more. XGB SKLearn from xgboost import XGBClassifier from sklearn import model selection from sklearn import metrics import lightgbm as lgb In [30]: train_df = train df.to pandas() test_df = test_df.to_pandas() using KFold to get best model split skf = model selection.StratifiedKFold(n splits=3, shuffle=True, random state=42) test predictions = [] oof predictions proba = np.zeros(len(train df)) X test = test df[train cols] for idx, (train idx, valid idx) in enumerate(skf.split(train df, train df[target col])): print("="*100) print("FOLD : ", idx) print(" "*100) X train = train df.iloc[train idx][train cols] y train = train df.iloc[train idx][target col] X valid = train df.iloc[valid idx][train cols] y valid = train df.iloc[valid idx][target col] print("Trian :", X train.shape, y train.shape) print("Valid :", X valid.shape, y valid.shape) XGB params = { 'n estimators': 14140, 'learning rate': 0.0111538499996725174, 'max depth': 4, 'reg alpha': 2.9735993984217104, 'reg lambda': 2.214625977393439, 'scale pos weight': 0.9, 'subsample': 0.7768428006197691, 'gamma': 0.5 #XGBoost model = XGBClassifier(**XGB params) model.fit(X_train, y_train, eval_set=[(X_valid, y_valid)], early stopping rounds=200, eval metric='auc', verbose=100) y valid pred = model.predict proba(X valid)[:, 1] predictions = model.predict proba(X test)[:, 1] test predictions.append(predictions) oof predictions proba[valid idx] = y valid pred score = metrics.roc auc score(train df[target col], np.asnumpy(oof predictions proba)) test predictions = np.mean(np.asnumpy(test predictions), axis=0) print("OOF MEAN AUC :", score) ss df['target'] = test predictions ss df.to csv("submission 3 xgb.csv", index=False) ss df.head() Score of XGB Classifier Sklearn Leaderboard C Refresh YOUR RECENT SUBMISSION Score: 0.78698 submission_3_xgb.csv Submitted by Rakshit Garg \cdot Submitted just now Private score: Jump to your leaderboard position LGBM Classifier using KFold to get best model split In [52]: skf = model selection.StratifiedKFold(n splits=5, shuffle=True, random state=42) test predictions = [] oof_predictions_proba = np.zeros(len(train_df)) X test = test df[train cols] for idx, (train idx, valid idx) in enumerate(skf.split(train df, train df[target col])): print("="*100) print("FOLD : ", idx) print(" "*100) X train = train df.iloc[train idx][train cols] y train = train df.iloc[train_idx][target_col] X valid = train df.iloc[valid idx][train cols] y valid = train df.iloc[valid idx][target col] print("Train :", X train.shape, y train.shape) print("Valid :", X valid.shape, y valid.shape) params = { 'boosting type': 'gbdt', 'tree learner': 'feature', #''serial' or feature' or 'data' or 'voting' 'num leaves': 31, 'max depth': -1, 'learning rate': 5e-2, 'n estimators': 10000, 'importance type': 'gain', 'subsample for bin': 200000, 'objective': 'binary', 'min split gain': 0.0, 'min child weight': 1e-3, 'min child samples': 20,

#'bagging freq': 0,

'reg_alpha': 0.2,
'reg_lambda': 0.2,
'random state': 42,

'n jobs': -1,

#'bagging_fraction': 1.0,
#'feature fraction': 1.0,

model = lgb.LGBMClassifier(**params)

eval metric='auc',

test predictions.append(predictions)

ss df.to csv("submission 4 lgb.csv", index=False)

submission_7_lgb.csv

Jump to your leaderboard position

Submitted by Rakshit Garg · Submitted just now

verbose=100)

print("OOF MEAN AUC :", score)

In [55]: ss df['target'] = test predictions

Score of LGBM Classifier Sklearn

YOUR RECENT SUBMISSION

Leaderboard

ss df.head()

early stopping rounds=200,

y_valid_pred = model.predict_proba(X_valid)[:, 1]
predictions = model.predict_proba(X_test)[:, 1]

oof predictions proba[valid idx] = y valid pred

test predictions = np.mean(np.asnumpy(test predictions), axis=0)

model.fit(X_train, y_train, eval_set=[(X_valid, y_valid)],

score = metrics.roc auc score(train df[target col], np.asnumpy(oof predictions proba))

C Refresh

Score: 0.78544

Private score:

Rakshit_102083022_CO28

Problem Name- Netflix Appetency

Problem Type- Classification

Problem Link- https://www.kaggle.com/c/netflix-appetency/overview

Lab Evaluation