Republic of Yemen
IBB University
Faculty of Science
Department of IT
Data Mining

الجمهورية اليمنية
جامعة إب
كلية العلوم التطبيقية
قسم : تقنية معلومات
مقرر : تنقيب البيانات

الطلاب التالية أسمائهم

- ايمن محمد ناجي قمحان
- ضياء فضل الحضرمي
- حازم هزام العمري

DEMAND FORECASTING

Mission is to create a 3-month demand forecasting model for the relevant store chain using the following time series and machine learning techniques:

Random Noise

Lag Shifted Features

Rolling Mean Features

Exponentially Weighted Mean Features

Custom Cost Function

Model Validation with LightGBM

Solving the problem of multiple unclear products in some markets and some products required by customers are not available. This model predicts the impact of products on purchasing in a shopping period and a decision is made to purchase the most demanded products and reduce the demand for those with scarce demand.

Dataset is here: https://www.kaggle.com/c/demand-forecasting

Dataset Description

The objective of this competition is to predict 3 months of item-level sales data at different store locations.

File descriptions

train.csv - Training data

test.csv - Test data (Note: the Public/Private split is time based)

sample_submission.csv - a sample submission file in the correct format

Data fields

date - Date of the sale data. There are no holiday effects or store closures.

store - Store ID

item - Item ID

sales - Number of items sold at a particular store on a particular dat

In Arabic language

الهدف من المشروع كالتالي :

التنبؤ بالطلب للمتاجر

المهمة هي إنشاء نموذج تنبؤ بالطلب لمدة 3 أشهر لسلسلة المتاجر المعنية باستخدام السلاسل الزمنية وتقنيات التعلم الآلي التالية:

الضوضاء العشوائية

خصائص التأخير المُحوّل

خصائص المتوسط المنحدر

خصائص المتوسط المرجح أسيًا

دالة التكلفة المُخصصة

التحقق من صحة النموذج باستخدام LightGBM

حل مشكلة وهي تعدد المنتجات غير المرغوبة في بعض المتاجر وبعض المنتجات المطلوبة لدى العملاء لا يتم توفيرها فهذا المودل يتنبئ في المنتجات الأثر شراءً في المتجر في مده معينة ومنة يتم اتخاذ قرار لشراء المنتجات الأكثر طلباً وتقليل الطلب للمنتجات التي لديها طلب شحيح

مجموعة البيانات متوفرة هنا: https://www.kaggle.com/c/demand-forecasting

وصف مجموعة البيانات

الهدف من هذه المسابقة هو التنبؤ ببيانات مبيعات المنتجات على مدار ثلاثة أشهر في مواقع متاجر مختلفة.

وصف الملفات

train.csv - بيانات التدريب

test.csv - بيانات الاختبار (ملاحظة: التقسيم بين عام وخاص يعتمد على الوقت)

sample_submission.csv - نموذج لملف الإرسال بالتنسيق الصحيح

حقول البيانات

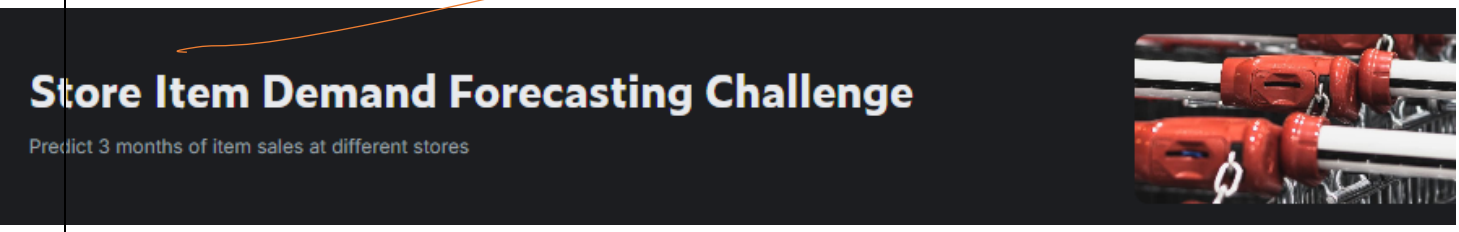
date - تاريخ بيانات البيع. لا توجد عطلات أو إغلاقات للمتاجر.

store - معرف المتجر

item - معرف المنتج

sales - عدد المنتجات المباعة في متجر معين في تاريخ محدد.

هنـا تم جلب البيانات



```python
# HELPER FUNCTIONS (UTILS)

# Check dataframe
def check_df(dataframe, head=5, tail=5, quan=False):
    print("##################### Shape #####################")
    print(dataframe.shape)
    print("##################### Types #####################")
    print(dataframe.dtypes)
    print("##################### Head #####################")
    print(dataframe.head(head))
    print("##################### Tail #####################")
    print(dataframe.tail(tail))
    print("##################### NA #####################")
    print(dataframe.isnull().sum())

    if quan:
        print("##################### Quantiles #####################")
        print(dataframe.quantile([0, 0.05, 0.50, 0.95, 0.99, 1]).T)

# Date Features
def create_date_features(df):
    df['month'] = df.date.dt.month
    df['day_of_month'] = df.date.dt.day
    df['day_of_year'] = df.date.dt.dayofyear
    df['week_of_year'] = df.date.dt.weekofyear
    df['day_of_week'] = df.date.dt.dayofweek
    df['year'] = df.date.dt.year
    df["is_wknd"] = df.date.dt.weekday // 4
    df['is_month_start'] = df.date.dt.is_month_start.astype(int)
    df['is_month_end'] = df.date.dt.is_month_end.astype(int)
    return df

# Random Noise
def random_noise(dataframe):
    return np.random.normal(scale=1.6, size=(len(dataframe),))

# Lag/Shifted Features
def lag_features(dataframe, lags):
    for lag in lags:
        dataframe['sales_lag_' + str(lag)] = dataframe.groupby(["store", "item"])['sales'].transform(
            lambda x: x.shift(lag)) + random_noise(dataframe)
    return dataframe

# Rolling Mean Features
def roll_mean_features(dataframe, windows):
    for window in windows:
        dataframe['sales_roll_mean_' + str(window)] = dataframe.groupby(["store", "item"])['sales']. \
                                                      transform(
            lambda x: x.shift(1).rolling(window=window, min_periods=10, win_type="triang").mean()) + random_noise(
            dataframe)
    return dataframe

# Exponentially Weighted Mean Features
def ewm_features(dataframe, alphas, lags):
    for alpha in alphas:
        for lag in lags:
            dataframe['sales_ewm_alpha_' + str(alpha).replace(".", "") + "_lag_" + str(lag)] = \
                dataframe.groupby(["store", "item"])['sales'].transform(lambda x: x.shift(lag).ewm(alpha=alpha).mean())
    return dataframe

# Custom Cost Function
def smape(preds, target):
    n = len(preds)
    masked_arr = ~((preds == 0) & (target == 0))
    preds, target = preds[masked_arr], target[masked_arr]
    num = np.abs(preds - target)
    denom = np.abs(preds) + np.abs(target)
    smape_val = (200 * np.sum(num / denom)) / n
    return smape_val

def lgbm_smape(preds, train_data):
    labels = train_data.get_label()
    smape_val = smape(np.expm1(preds), np.expm1(labels))
    return 'SMAPE', smape_val, False

# Feature Importance
def plot_lgb_importances(model, plot=False, num=10):

    gain = model.feature_importance('gain')
    feat_imp = pd.DataFrame({'feature': model.feature_name(),
                             'split': model.feature_importance('split'),
                             'gain': 100 * gain / gain.sum()}).sort_values('gain', ascending=False)
    if plot:
        plt.figure(figsize=(10, 10))
        sns.set(font_scale=1)
        sns.barplot(x="gain", y="feature", data=feat_imp[0:25])
        plt.title('feature')
        plt.tight_layout()
        plt.show()
    else:
        print(feat_imp.head(num))

# Kaggle input part
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

ات من منصة Kaggel

يقوم هذا الكود بتهيئة بيئة العمل الخاصة بتحليل البيانات والتعلم الآلي عبر استيراد المكتبات التالية:

- numpy , pandas لتحليل البيانات ومعالجتها.
- matplotlib, seaborn للرسم البياني.
- Lightgbm لبناء نموذج تعلم آلي فعال.
- LabelEncoder لتحويل البيانات الفئوية إلى عددية.
- إعدادات عرض pandas وتحكم في رسائل التحذير لتحسين تجربة المستخدم.

```python
# Import necessary libraries and make necessary arrangements
import time
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import lightgbm as lgb
import warnings
from sklearn.preprocessing import LabelEncoder

pd.set_option('display.max_columns', None)
pd.set_option('display.width', 500)
warnings.filterwarnings('ignore')
```

*دالة لفحص وتحليل أولي لبيانات DataFrame تعرض التالي:

- الشكل العام للبيانات عدد الصفوف، الأعمدة
- أنواع البيانات لكل عمود
- أول وأخر عدد معين من الصفوف (افتراضي 5)
- عدد القيم المفقودة في كل عمود
- إذا تم تفعيل quan=True، تُعرض أيضًا القيم المئوية(Quantiles)

```python
def check_df(dataframe, head=5, tail=5, quan=False):

```

* تُستخدم لاستخراج **ميزات مشتقة من التاريخ** من عمود date، مثل:

- الشهر، اليوم من الشهر، اليوم من السنة
- الأسبوع، اليوم من الأسبوع، السنة
- هل التاريخ في بداية أو نهاية الشهر؟
- هل اليوم عطلة (نهاية أسبوع)؟

هذه الميزات تساعد النماذج على فهم الأنماط الزمنية.

```python
def check_df(dataframe, head=5, tail=5, quan=False):

```

```python
def random_noise(dataframe):

```

* تُضيف **ضجيج عشوائي** (Noise) إلى البيانات، وهو أمر مفيد لمنع النماذج من التعلّم الزائد(Overfitting)

```python
def lag_features(dataframe, lags):

```

تُنشئ ميزات **الإزاحة الزمنية (Lag)** للمبيعات السابقة، مثل مبيعات اليوم السابق، أو قبل يومين، إلخ.

- تقوم بذلك لكل مجموعة (store, item)
- تُستخدم هذه الميزات لتعليم النموذج العلاقة بين مبيعات اليوم والأيام السابقة

```python
def roll_mean_features(dataframe, windows):

```

* تُنشئ ميزات **المتوسط المتحرك (Rolling Mean)** باستخدام نافذة windowمعينة.

- مفيدة لتحليل الاتجاهات على مدى زمني معين.
- تُستخدم دالة rolling() مع نوع نافذة مثل"tiring"
.

```python
def ewm_features(dataframe, alphas, lags):

```

* تُنشئ ميزات **المتوسط المرجح أسيًا(EWM)** ، الذي يعطي وزنًا أكبر للبيانات الأحدث.

- alphaيتحكم بمقدار الوزن للبيانات الحديثة.
- تُطبق لكل مجموعة (store, item) على بيانات المبيعات.

* دالة لحساب **SMAPE - Symmetric Mean Absolute Percentage Error**، وهي مقياس دقيق لتقييم النماذج التي تتنبأ بقيم مستمرة، خصوصًا في السلاسل الزمنية.

- ❖ تعطي نتائج بين 0 و100 كلما اقتربت من 0 كان النموذج أفضل

```
def smape(preds, target):

```

* تُستخدم لربط SMAPE بنموذج LightGBM كتقييم مخصص (custom metric) أثناء التدريب.

- ❖ تُحوّل القيم من Log إلى طبيعتها قبل التقييم باستخدام np.expm1()

```
def lgbm_smape(preds, train_data):

```

*تعرض أو تطبع **أهمية الميزات (Feature Importance)** في نموذج.LightGBM

- ❖ تعتمد على طريقتين: عدد الانقسامات (split) أو إجمالي التأثير.(gain)
- ❖ عند تفعيل plot=Trueترسم رسمًا بيانيًا لأهم 25 ميزة.

```
def plot_lgb_importances(model, plot=False, num=10):

```

جزء قراءة البيانات من كاجل

يُستخدم للحصول على قائمة بكل الملفات الموجودة في مجلد /kaggle/input.

- ❖ مفيد عند استخدام منصة Kaggle للتأكد من الملفات المتاحة للقراءة.

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

- هذا القسم يقوم بتحميل البيانات وتحضيرها للمعالجة.
- تم دمج بيانات التدريب والاختبار لتسهيل معالجة موحدة لجميع البيانات .
- لاحقًا سيتم فصلهم مرة أخرى عند بناء النموذج والتنبؤ.

```
df = pd.concat([train, test], sort=False)
```

التحليل والاستكشاف الأولي

*نتحقق من **نطاق التواريخ** في كل من البيانات الكاملة (df) والتدريب (train) والاختبار.(test)

```
df["date"].min(), df["date"].max()
train["date"].min(), train["date"].max()
test["date"].min(), test["date"].max()
```

- نستخدم الدالة check_df التي شرحناها سابقًا لفحص كل مجموعة بيانات: عدد الأعمدة، القيم المفقودة، أنواع البيانات، إلخ.

```
check_df(train)
check_df(test)
check_df(df)
```

- نُحلل توزيع المبيعات باستخدام إحصائيات مئوية لتحديد القيم المتطرفة والوسيط.

```
df["sales"].describe([0.10, 0.30, 0.50, 0.70, 0.80, 0.90, 0.95, 0.99])
```

معلومات حول المتاجر والمنتجات

- ❖ نحصل على:

- ❖ عدد المتاجر الفريد
- ❖ عدد العناصر الفريدة
- ❖ عدد العناصر المتاحة في كل متجر

```
df[["store"]].nunique()
df[["item"]].nunique()
df.groupby(["store"])["item"].nunique()
```

- ❖ تحليل المبيعات حسب كل متجر وعُنصر: مجموع، متوسط، وسيط، وانحراف معياري.

```
df.groupby(["store", "item"]).agg({"sales": ["sum", "mean", "median", "std"]})
```

استخراج ميزات تاريخية

- ❖ نحول العمود date إلى صيغة زمنية، ثم نستخدم دالة create_date_features لاستخراج:

- ❖ الشهر، الأسبوع، السنة، بداية/نهاية الشهر، إلخ.

```
df['date'] = pd.to_datetime(df['date'])
df = create_date_features(df)
check_df(df)
```

- ❖ تحليل شهري للمبيعات حسب المتجر والعنصر.

```
df.groupby(["store", "item", "month"]).agg({"sales": ["sum", "mean", "median", "std"]})
```

ميزات الإزاحة الزمنية (Lag Features)

❖ بعد فرز البيانات زمنيًا، ننشئ ميزات للمبيعات السابقة (Lagged Sales)، وهي مهمة جدًا للتنبؤ بناءً على القيم السابقة.

```
df.sort_values(by=['store', 'item', 'date'], axis=0, inplace=True)
df = lag_features(df, [91, 98, 105, 112, 119, 126, 182, 364, 546, 728])
```

المتوسطات المتحركة

❖ نُضيف ميزات للمتوسطات المتحركة طويلة المدى (سنة، سنة ونصف)، مما يساعد على التقاط الاتجاهات.

```
df = roll_mean_features(df, [365, 546])
```

المتوسطات الأسية المرجحة (EWM)

❖ تُنشأ ميزات تأخذ في الاعتبار الأوزان الزمنية للأسابيع أو الأشهر الماضية، حيث تُعطى أهمية أكبر للقيم الأحدث.

```
alphas = [0.95, 0.9, 0.8, 0.7, 0.5]
lags = [91, 98, 105, 112, 180, 270, 365, 546, 728]
df = ewm_features(df, alphas, lags)
```

كود التحضير للنموذج

❖ نحول المتغيرات الفئوية (store, item, ...) إلى تمثيل رقمي باستخدام **One-Hot Encoding**.

❖ نحول المبيعات إلى **القيمة اللوغاريتمية** لتقليل تأثير القيم المتطرفة.

```
df = pd.get_dummies(df, columns=['store', 'item', 'day_of_week', 'month'])
df['sales'] = np.log1p(df["sales"].values)
```

-* نقسم البيانات زمنياً للتدريب :

❖ مجموعة **التدريب** حتى نهاية 2016

❖ مجموعة **التحقق** هي أول 3 شهور من 2017، لمحاكاة التنبؤ في 2018

```
train = df.loc[(df["date"] < "2017-01-01"), :]
val = df.loc[(df["date"] >= "2017-01-01") & (df["date"] < "2017-04-01"), :]
```

+- نحدد المتغير المستهدف (sales) والمتغيرات المدخلة (cols
يجب أن تحتوي على أسماء الأعمدة التي تم استخراجها.)

```
Y_train = train['sales']
X_train = train[cols]
Y_val = val['sales']
X_val = val[cols]
```

* نتأكد من صحة أبعاد البيانات قبل التدريب.

```
Y_train.shape, X_train.shape, Y_val.shape, X_val.shape
```

- إعدادات نموذج:LightGBM

- ❖ mae مقياس الخطأ
- ❖ num_leaves, max_depth تتحكم بتعقيد النموذج
- ❖ early_stopping_rounds توقف التدريب عند عدم التحسن
- ❖ learning_rate معدل التعلم الصغير لتجنب التذبذب

```
lgb_params = {
    'metric': {'mae'},
    'num_leaves': 10,
    'learning_rate': 0.02,
    'feature_fraction': 0.8,
    'max_depth': 5,
    'verbose': 0,
    'num_boost_round': 1000,
    'early_stopping_rounds': 200,
    'nthread': -1
}
```

- ❖ إعداد بيانات التدريب والتحقق لاستخدامها في نموذج LightGBM.

```
lgbtrain = lgb.Dataset(data=X_train, label=Y_train, feature_name=cols)
lgbval = lgb.Dataset(data=X_val, label=Y_val, reference=lgbtrain, feature_name=cols)
```

```python
model = lgb.train(
    lgb_params,
    lgbtrain,
    valid_sets=[lgbtrain, lgbval],
    num_boost_round=lgb_params['num_boost_round'],
    callbacks=[lgb.early_stopping(lgb_params['early_stopping_rounds']), lgb.log_evaluation(100)],
    feval=lgbm_smape
)

```

❖ تدريب النموذج باستخدام:

❖ التحقق المبكر(Early stopping)
❖ تقييم مخصص عبرSMAPE
❖ طباعة النتائج كل 100 جولة

```python
y_pred_val = model.predict(X_val, num_iteration=model.best_iteration)
smape(np.expm1(y_pred_val), np.expm1(Y_val))

```

تقييم النموذج على مجموعه التحقق

❖ نقوم بالتنبؤ على مجموعة التحقق (X_val) باستخدام أفضل تكرار من التدريب(best_iteration)
* ثم نحسب SMAPE على القيم الأصلية (بعد عكس log1p باستخدام expm1) لتقييم دقة النموذج.

```python
plot_lgb_importances(model, num=30, plot=True)

```

عرض اهم الميزات

❖ نستخدم دالة plot_lgb_importances لرسم أهمية الميزات في النموذج النهائي، مما يساعدنا على فهم ما يؤثر أكثر على التنبؤ.

تدريب النموذج النهائي

❖ يتم استخدام **كامل بيانات التدريب** لتدريب النموذج النهائي، واستبعاد الصفوف التي لا تحتوي على sales)وهي مجموعة الاختبار.(

```
train = df.loc[~df.sales.isna()]
test = df.loc[df.sales.isna()]
```

❖ نستخدم **أفضل عدد تكرارات** تم الحصول عليه سابقًا لتدريب نموذج قوي على كامل البيانات.

```
lgb_params = {..., "num_boost_round": model.best_iteration}
model = lgb.train(lgb_params, lgbtrain_all, num_boost_round=model.best_iteration)
```

إنشاء ملف التقديم Submission

❖ نحضر الملف حسب تنسيق مسابقة كاجل:

- نستخدم sales gid
- نحول القيم من log إلى القيم الأصلية

```
submission_df = test.loc[:, ['id', 'sales']]
submission_df['sales'] = np.expm1(test_preds)
submission_df['id'] = submission_df.id.astype(int)
submission_df.to_csv('submission_demand.csv', index=False)
```

حفض النموذج وملف التقييم

نحفظ :

- النموذج المدرب بصيغة pkl.
- ملف النتائج بصيغة csv.

* يتم حفظ كل ذلك في **Google Drive** لاستخدامه لاحقًا أو تحميله.

```
with open('/content/drive/My Drive/lgbm_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)
submission_df.to_csv('/content/drive/My Drive/submission_demand.csv', index=False)
```

تحميل النموذج نخليه لاحقاً

- نستخدم مكتبة pickle لإعادة تحميل النموذج عند الحاجة دون إعادة تدريبه.

```
with open('/content/drive/MyDrive/lgbm_model.pkl', 'rb') as f:
    model = pickle.load(f)
```

التنبؤ اليدوي لمتجر معين

- تُجهز هذه الدالة إدخال يدوي لتنبؤ مبيعات منتج معين في متجر معين في تاريخ معين:
  - يتم استخراج خصائص التاريخ
  - يتم تنفيذOne-Hot Encoding
  - ملء الأعمدة الناقصة بـ 0
  - ترتيب الأعمدة تمامًا كما في التدريب

```
def prepare_input(store, item, date_str, feature_columns): ...
```

- يتم تمرير البيانات إلى النموذج للحصول على **توقع المبيعات لليوم المحدد.**

```
input_df = prepare_input(store=3, item=25, date_str='2018-01-01', feature_columns=cols)
prediction = model.predict(input_df)[0]
```

```
plt.bar(['Predicted Sales'], [prediction], color='green')
```

❖ يتم رسم **الرسم البياني** الذي يُظهر قيمة المبيعات المتوقعة بشكل مرئي أنيق.

خلاصة المشروع: توقع الطلب باستخدام LightGBM

- **قمنا ببناء نموذج دقيق لتوقع المبيعات اليومية لكل منتج في كل متجر باستخدام تقنيات تعلم الآلة.**
- **اعتمدنا على تحويل البيانات الزمنية إلى ميزات ذكية مثل:**
  - **الإزاحات الزمنية (Lag Features)**
  - **المتوسطات المتحركة (Rolling Mean)**
  - **المتوسطات الأسية المرجحة (EWM)**
  - **مؤشرات التاريخ (الشهر، اليوم، بداية/نهاية الشهر...إلخ)**
- **استخدمنا نموذج LightGBM نظراً لسرعته ودقته مع البيانات الكبيرة.**
- **تم تقييم النموذج باستخدام مقياس SMAPE لضمان جودة التنبؤ.**
- **بعد التحقق درّبنا نموذجاً نهائيًا على كامل البيانات وولّدنا ملف التقديم النهائي.**
- **أنشأنا دالة للتنبؤ اليدوي بأي تاريخ/منتج/متجر، مع عرض النتائج بطريقة رسومية بسيطة.**

النتيجة ...

نموذج فعال قادر على دعم اتخاذ القرار في التخطيط للمخزون، التسعير، وإدارة الطلبات بكفاءة عالية.

اشراف المهندس / فؤاد الـــدغار