9-dars uchun Report

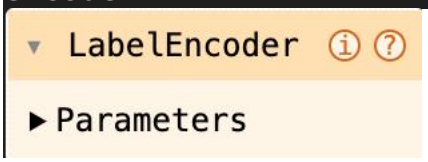
10-darsda biz **ML Model Structure** haqida gaplashdik unda: D**ata Collection, Data bilan tanishuv va Data Preprocessing**larni tugatganimiz va so'ngra quyida qolgan qismlari bular; **Model (Algorithm) Selection, Model Training va Evaluation**larni ko'rib chiqdik.

Model va algoritmga tushunarliroq tarif berish o'laroq uni yaxshiroq hayotiy o'rinda tushunishga harakat qildik masalan Modelni bir ovqat desak, algoritm bu retsept ya'ni ovqatni qancha vaqtda pishirish, qancha va nima qo'shish, qanday ketma-ketlikda bajarish, qancha qovurish va hokazolar. Yoki o'qituvchi bilimi bu model lekin uni o'quvchiga moslashi ya'ni o'yin tarzida, video ko'rish orqali yoki og'zaki imtixon qilish orqali yoki jamolarga bo'lib o'rgatishni misol qilib aytsak bo'ladi algoritm uchun.

ML algorithm family (Supervised ML) mavjud bo'lib aslida buni biz o'zimiz yaxshiroq eslab qolish uchun oila tarziga keltiib oldik. Bunda **Supervised ML** algoritmlar juda ko'p bo'lib, ularni kodlari bilan eslab qolish mushkul ishdir. Shuning uchun biz ularni kutubxonalaridan chaqirishim ancha qulaydi.
Bularga; **Linear (Linear Regression, Logistic Regression), Tree-based (Decision Tree, Random Forest), Distance-based(KNN, SVM) va Ensemble(Random Forest, Gradient Boosting)** lar kiradi. Data Preprocessing orqali tayyorlab olgan datasetimizni kompyuterga tuhsunarli qilib o'rgatish albatta har hil natijalar beradi. Loyihamizda kelgusida '**Comperative Evaluation'** qismi bo'ladi unda bir nechta algoritmlar bilan ishlatib ko'rib unga baho beramiz. Va o'sha algoritm bilan model yasaymiz, proyekt quramiz.

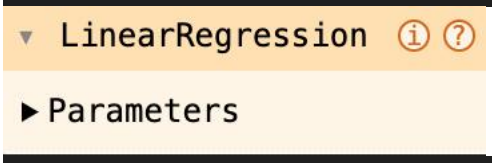
Quyida kodlardan parchlar keltirilgan;

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder
```




```
from sklearn.linear_model import LinearRegression,
LogisticRegression # bu yerda LinearRegression bu faqat
Regression uchn ishlaydi, LogisticRegression esa faqat
Classification uchun ishlaydi.

lr = LinearRegression()
lr
```




```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestRegressor
```

Liner Regression bu baseline modellar hisoblanadi. Ya'ni bir boshidan baseline model qurib olib, kuchsizroq model qurib keyin uni rivojlantirishga harakat qilamiz, Ya'ni o'zimi bir planka qo'yamiz va plankadan oshishga harakat qilamiz.

Bunda Classification-> Logistic Regression (mantiq funksiyasi, asosida ishlaydi masalan (0,1) ya'ni discrete - chekli 2 ta qiymatdan iborat), Regression-> Linear Regressiondir.

Logistic Regression - bu classification uchun ishlatilinadi va asosan binary classification'larda yaxshi ishlaydigan algoritm hisoblanadi. Bunda e'tibor berishimiz kerak bo'lgan 3 jihat bor; Logistic Regression, classification va Regression uchun ishlamasligi.

Classification tasklar uchun effektiv algoritmlardan biri; Bunda sodda (ya'ni unchalik katta yoki kompleks bo'magan loyihalarda ishlatsak ham bo'ladi), kichik datasetlarda yaxshi ishlaydi, va Binary classificationda yashi ishlaydi (2 ta klasslarda iborat bo'lganda)

Logistic Regressiondagi bosqichlar quyidagilardan iborat; Data Preprocessing, (x,y), Splitting (train,test), Training(fitting), Predicting va Evaluating.

Model trainingda umumiy scaling jarayonida quyida formula orqali bajariladi; $\mathbf{x_{scaled}} = \frac{x - xmin}{xmax - xmin}$. Bazida bu holatda garchi ustunlar int bo'lsa ham scaling qilishda floatga o'tib qolishi mumkin.

Keyingi bosqichlarda esa biz ma'lumotlarni ikkiga ajratiib olishimiz kerak. Bamisoli o'qituvchi o'quvchilarga imtixonga tayyrolash uchun asosiy bilimlarni ko'p beradi va imtixonda esa bergan bilimiga qaranga ancha kam hajda uni imtixon qilib oladi. Bunda Train uuchun datasetimizdan kelib chiqib 70% yoki 80% olamiz, Test uchun esa 30% yoki 20% olamiz. Va bu aynan Splitting deyiladi. Aslida Splitting qilish bir muncha qiyin va vaqt olganligi sababli tayyor kutubxonadan fodalanish bizga ancha qulaylik keltiradi.

```
from sklearn.model_selection import train_test_split #
buning asosiy vazifasi 2 ga ajratib berish
# inputlarni olib qolishni oson yo'li bu outputni drop
qilish yetadi.

x = df.drop(columns=['annual_cost_healthy_diet_usd'])
x = df.drop(columns=['country_code']) # bu esa
datasetimizda precit uchun muhim bo'lmagan ustundan
qutulish mumkin
```

```
y = df['annual_cost_healthy_diet_usd'].astype(int) # buni
qilishdan maqsad keyinchalik x_test qilganda bizga
chiqaradigan qiymatimiz float emas int chiqqani
maqulroqligi sabab

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42) # shu xolatda
random_state=42 modelni balansli ishlashi uchun.
```

Bu yerda test_size =0.2 degani 20% berildi qolgani avtmotik tarzda trainga beraman degani.

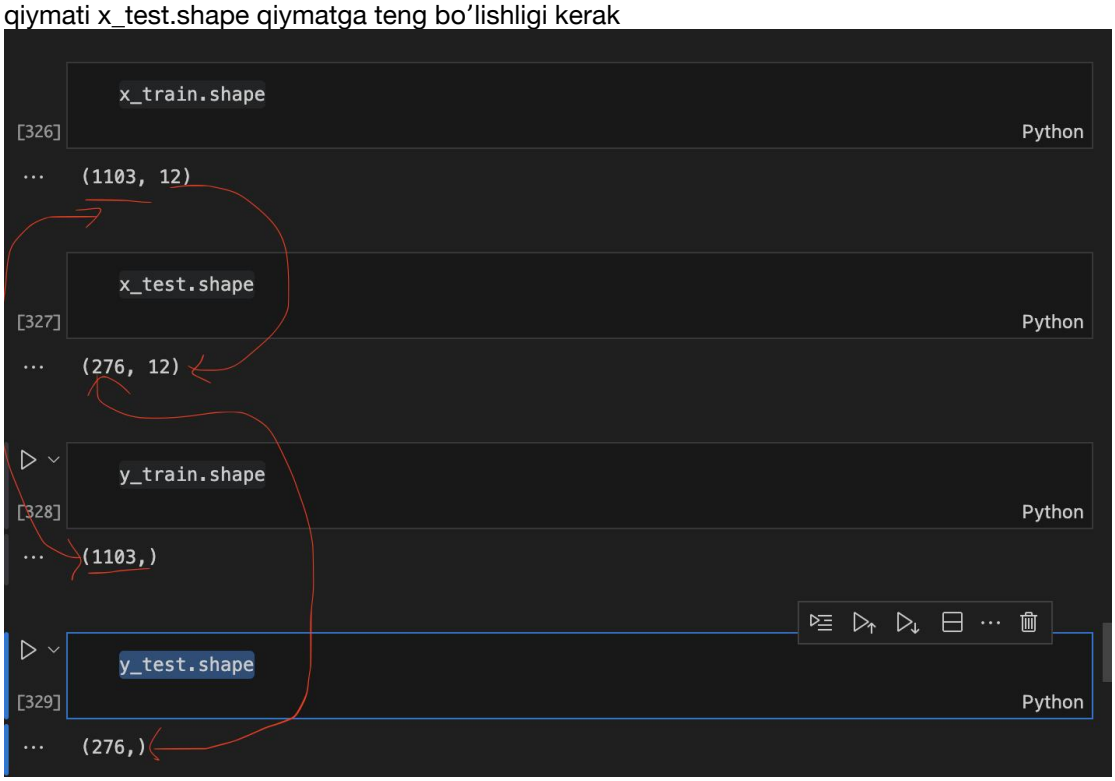
```
x_train.shape
```

```
x_test.shape
```

```
y_train.shape
```

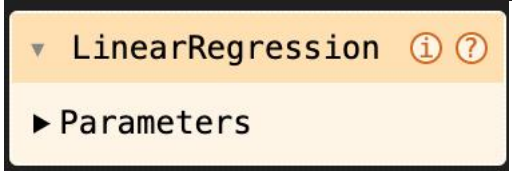
```
y_test.shape
```

Bunda es ax_train.shape qiymati y_train.shape bilan bir xil va x_train.shapening ikkinchi qiymati x_test.shape qiymatga teng bo’lishligi kerak



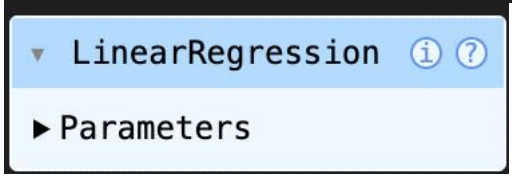

```
from sklearn.linear_model import LinearRegression
```

```
lin_reg = LinearRegression()
lin_reg
```



Bunda not fitted bo’lgan edi ya’ni hali ishga tusmaganligini bildiradi.

```
lin_reg.fit(x_train,y_train) # bu yerda traininglar bilan train ishlatilinadi.
```



Bunda esa rangi ko’k bo’ldi bu esa fitted yani ishga tushirilganini bildiradi.

```
y_pred = lin_reg.predict(x_test) # aqilli qilib olgan testimizni predict qilamiz
y_pred[1:10]
```

```
array([-0.00064414, -0.0171581 ,  0.00195075, -0.00100796,  0.00553704,
       -0.00094837,  0.00534536, -0.00096763, -0.00038961])
```

Endi modelimiz qanchalik yaxshi ishlashini qayerdan bilamiz? Bunianiqlash uchun bizga metrics kerak bo'ladi. Ya'ni metrics orali baholaymiz modelimiz yaxshi ishlayaptimi yoki yo'q.

Mening datasetimiz LinearRegreesionda ishlagani sabab men mean_absolute_error orqali tekshirdim. Agar Logistic Regression bo'lganida biz accruacy_score olgan bo'lar edik.

```
from sklearn.metrics import accuracy_score
```

Menda 0.00579.. atrofida chiqdi bu holatda **mean_absolute_error** qanchalik past chiqsa shunchalik yaxshi ekan.

```
mean_abs_error
```

```
0.0057918175403781896
```

s