گزارش پروژه پیش بینی درآمد افراد

(Income Prediction Project)

هدف پروژه

هدف ما در این پروژه، پیش‌بینی این است که آیا درآمد یک فرد بیش از K50 دلار در سال است یا خیر.

این مسئله یک مسئله‌ی طبقه‌بندی دودویی (Binary Classification) است که با استفاده از داده‌های جمعیتی (سن، تحصیلات، شغل، وضعیت تأهل، جنسیت و ...) انجام می‌شود.

بارگذاری داده‌ها

ابتدا داده‌ها را بارگذاری کرده و به دو بخش ویژگی‌ها (X) و برچسب (y) تقسیم کردیم.

```
train_data = pd.read_csv('train.csv')
X = train_data.drop("income", axis=1)
y = train_data["income"]
```

تقسیم داده‌ها به آموزش و تست

برای اینکه مدل بتواند تعمیم پیدا کند، داده‌ها را به دو بخش آموزش و تست تقسیم کردیم.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
```

حذف ستون (Fnlwgt)

```
X_train = X_train.drop(columns=["fnlwgt"], errors="ignore")
```

- این خط ستون fnlwgt رو از داده‌ها حذف می‌کنه.
- ستون fnlwgt (Final Weight) در دیتاست Adult Income وجود داره، اما معمولاً در پیش‌بینی درآمد فرد نقش خیلی زیادی نداره چون یک وزن آماریه که برای مقیاس‌گذاری نمونه‌ها در نظر گرفته شده، نه یک ویژگی شخصی مستقیم.
- به همین دلیل اغلب در تحلیل‌های یادگیری ماشین این ستون حذف می‌شه.
- پارامتر errors="ignore" باعث می‌شه اگر به هر دلیلی این ستون توی داده وجود نداشت، کد ارور نده

جایگزینی مقادیر گمشده

```
for col in X_train.select_dtypes(include=["object"]):
    X_train[col] = X_train[col].replace("?", X_train[col].mode()[0])
```

- توی دیتاست، بعضی از ستون‌های دسته‌ای (categorical) مثل workclass، occupation، native-country دارای مقدار "?" هستن که نشونه‌ی داده‌ی گمشده (missing value) هست.
- این کد روی تمام ستون‌های دسته‌ای (متنی) اجرا می‌شه.
- برای هر ستون:
  - مقدار "?" رو پیدا می‌کنه.
  - بعد اون رو با مد (mode) ستون جایگزین می‌کنه.
  - مد همون مقداریه که بیشترین تکرار رو توی اون ستون داره

انتخاب ستون‌های دسته‌ای (Categorical Columns)

```
cat_cols = X_train.select_dtypes(include=["object"]).columns
```

- این خط میاد توی داده‌های X_train، تمام ستون‌هایی که نوعشون object (یعنی رشته‌ای/متنی) هست رو پیدا می‌کنه.
- نتیجه‌ی این خط یک لیست از نام ستون‌های متنیه

ساخت یک دیکشنری برای ذخیره‌ی انکودرها

```
le_dict = {}
```

- اینجا یک دیکشنری خالی تعریف شده که بعداً LabelEncoder هر ستون داخلش ذخیره می‌شه.
- دلیلش اینه که اگر بعداً خواستیم روی داده‌های تست یا داده‌های جدید هم همین کدگذاری رو انجام بدیم، بتونیم دقیقاً از همون انکودر استفاده کنیم (تا نگاشت مقادیر به عدد یکی باشه)

حلقه روی ستون‌های دسته

```
for col in cat_cols:
    le = LabelEncoder()
    X_train[col] = le.fit_transform(X_train[col])
    le_dict[col] = le
```

- برای هر ستون دسته‌ای (col):
  1. یک شیء LabelEncoder ساخته می‌شه.
  2. با دستور fit_transform تمام مقادیر متنی اون ستون به اعداد تبدیل می‌شن.

- مثلاً ستون education که شامل مقادیری مثل:
  - Bachelors
  - Masters
  - HS-grad
  - Doctorate

بود، تبدیل می‌شه به اعداد:

- Bachelors → 0
- Doctorate → 1
- HS-grad → 2
- Masters → 3

3. همون LabelEncoder در دیکشنری le_dict ذخیره می‌شه تا نگاشت مقادیر هر ستون حفظ بشه.

ایجاد ویژگی جدید (Features engineering)

```
for df_ in [X_train, X_test]:
    df_["capital_diff"] = df_["capital.gain"] - df_["capital.loss"]
    df_["is_over_40_hours"] = (df_["hours.per.week"] > 40).astype(int)
```

ویژگی جدید capital_diff

- ستون‌های capital.gain و capital.loss میزان سود و زیان سرمایه‌ی هر فرد رو نشون می‌دن.
- این خط کدی میاد تفاوت این دو رو حساب می‌کنه و داخل یک ویژگی جدید به اسم capital_diff ذخیره می‌کنه.
  - به جای بررسی جداگانه‌ی دو ستون، یک نمای کلی از وضعیت خالص سرمایه فرد به مدل می‌دیم.

ویژگی جدید
is_over_40_hours

- ستون hours.per.week نشون می‌ده فرد چند ساعت در هفته کار می‌کنه.
- این خط بررسی می‌کنه آیا فرد بیشتر از ۴۰ ساعت در هفته کار می‌کنه یا نه.
  - اگر بیشتر باشه ← مقدار 1
  - اگر ۴۰ یا کمتر باشه ← مقدار 0
  - ساعت کار طولانی می‌تونه نشونه‌ای از سطح درآمد بالاتر یا نوع شغل متفاوت باشه.
  - مدل با این ویژگی بهتر می‌تونه رابطه‌ی بین میزان کار و درآمد رو یاد بگیره

اعمال روی هر دو دیتافریم
(Train و Test)

- این حلقه باعث می‌شه هم روی داده‌های آموزشی (X_train) و هم روی داده‌های تست (X_test) همین ویژگی‌های جدید ساخته بشن.
- این کار خیلی مهمه چون مدل در مرحله‌ی پیش‌بینی هم باید دقیقاً همان ویژگی‌هایی رو داشته باشه که در آموزش داشته.

فراخوانی کتابخانه

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

- این کلاس از کتابخانه‌ی scikit-learn برای ساخت مدل درخت تصمیم استفاده می‌شه.
- درخت تصمیم یکی از الگوریتم‌های پرکاربرد در دسته‌بندی (Classification) هست که داده‌ها رو به صورت پرسش‌های متوالی (if/else) تقسیم می‌کنه.

تعریف مدل با تنظیمات خاص

```
dt = DecisionTreeClassifier(
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    criterion="gini",
    random_state=42
)
dt.fit(X_train, y_train)
```

- max_depth=10
  - حداکثر عمق درخت رو ۱۰ تعیین کرده.
  - جلوی بزرگ شدن بیش از حد درخت و overfitting رو می‌گیره.
- min_samples_split=5
  - حداقل تعداد نمونه‌هایی که برای تقسیم یک گره لازم هست = ۵.
  - یعنی اگر گره‌ای کمتر از ۵ نمونه داشته باشه، دیگه تقسیم نمی‌شه.
- min_samples_leaf=2
  - حداقل تعداد نمونه‌هایی که باید در هر برگ نهایی (leaf) باقی بمونن = ۲.
  - باعث می‌شه مدل برگ‌های خیلی کوچک (که فقط ۱ داده دارن) نسازه ← کاهش overfitting.
- criterion="gini"
  - معیار تقسیم گره‌ها شاخص جینی (Gini Impurity) هست.
  - شاخص جینی میزان ناپاکی (بی‌نظمی) داده‌ها رو تو هر گره اندازه می‌گیره.
- random_state=42
  - مقدار ثابت برای تکرارپذیری.
  - یعنی هر بار کد اجرا بشه، دقیقاً همون مدل ساخته می‌شه

ارزیابی مدل

```python
from sklearn.metrics import accuracy_score, classification_report, f1_score

# پیش‌بینی روی داده‌ی آموزش
y_train_pred = dt.predict(X_train)

# گزارش کامل
print(classification_report(y_train, y_train_pred))
```

- داده‌های آموزشی (X_train) به مدل داده می‌شن.
- خروجی پیش‌بینی شده در y_train_pred ذخیره می‌شه.
- حالا می‌تونیم پیش‌بینی‌ها رو با مقادیر واقعی (y_train) مقایسه کنیم.

```
              precision    recall  f1-score   support

       <=50K       0.89      0.95      0.92     15164
        >50K       0.80      0.62      0.70      4836

    accuracy                           0.87     20000
   macro avg       0.84      0.78      0.81     20000
weighted avg       0.86      0.87      0.86     20000
```

- Precision (دقت): از تمام پیش‌بینی‌های مثبت مدل، چند درصد درست بودن.
- Recall (بازخوانی): از کل موارد مثبت واقعی، چند درصد درست پیش‌بینی شدن.
- F1-score: میانگین موزون precision و recall → وقتی داده نامتوازن باشه، معیار مهمیه.
- Support: تعداد نمونه‌های واقعی در هر کلاس.

```
print("F1-score for >50K:", f1_score(y_train, y_train_pred, pos_label=">50K"))
```

- فقط امتیاز F1 برای کلاس ">50K" حساب می‌شه.
- این معیار نشون می‌ده مدل برای این کلاس چقدر خوب کار کرده