

هوش مصنوعی و سیستم های خبره

پروژه اول (درخت تصمیم)

فاطمه سادات باقری

400522355

1. توضیحات لازم کد :

```
class Node:
    def __init__(self , feature = None, threshold=None, left = None, right =
None,*, value= None ):
        self.feature = feature
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value

    def is_leaf_node(self):
        return self.value is not None
```

Node کلاس:

.این کلاس یک گره در درخت تصمیم را نمایش می دهد.

ویژگی های این کلاس عبارتند از:

.feature : ویژگی مربوط به تقسیم در گره (برای بچه ها نیاز نیست)

threshold: آستانه‌ی تقسیم برای ویژگی مذکور

left: زیردرخت چپ (بچه چپ) گره

right: زیردرخت راست (بچه راست) گره

value: اگر گره برگ باشد، اینجا مقدار پیش‌بینی شده برای این برگ ذخیره می‌شود.

is\_leaf\_node(): یک متد که بررسی می‌کند که آیا این گره برگ است یا نه.

```
class DecisionTree:
    def __init__(self, min_samples_split=2, max_depth=100, n_features=None):
        self.min_samples_split=min_samples_split
        self.max_depth=max_depth
        self.n_features=n_features
        self.root = None

    def fit(self, X, y):
        self.n_features = X.shape[1] if not self.n_features else min(X.shape[1],
self.n_features)
        self.root = self._grow_tree(X, y)

    def _grow_tree(self, X, y, depth=0):
        n_samples, n_feats= X.shape
        n_labels = len(np.unique(y))

        if (depth>= self.max_depth or n_labels==1 or
n_samples<self.min_samples_split):
            leaf_value = self._most_common_lable(y)
            return Node(value=leaf_value)

        feat_idx = np.random.choice(n_feats, self.n_features, replace=False)
```

```

        best_feature, best_thresh =self._best_split(X, y, feat_idxxs)

        left_idxxs, right_idxxs =self._split(X[:, best_feature ], best_thresh)

        left = self._grow_tree(X[left_idxxs, :], y[left_idxxs], depth+1)
        right = self._grow_tree(X[right_idxxs, :], y[right_idxxs], depth+1)

        return Node(best_feature, best_thresh, left, right)

def _best_split(self, X , y ,feat_idxxs):
    best_gain = -1
    split_idx, split_threshold = None, None

    for feat_idx in feat_idxxs:
        X_column = X[:, feat_idx]
        thresholds = np.unique(X_column)

        for thr in thresholds:
            gain = self._information_gain(y, X_column, thr)

            if gain > best_gain:
                best_gain = gain
                split_idx = feat_idx
                split_threshold = thr

    return split_idx, split_threshold

def _information_gain(self, y, X_column, threshold):
    parent_entropy = self._entropy(y)

    left_idxxs, right_idxxs = self._split(X_column, threshold)

    if len(left_idxxs) == 0 or len(right_idxxs) == 0:
        return 0

    n = len(y)
    n_l, n_r = len(left_idxxs), len(right_idxxs)
    e_l, e_r = self._entropy(y[left_idxxs]), self._entropy(y[right_idxxs])
    child_entropy = (n_l/n) * e_l + (n_r/n) * e_r

```

```

        information_gain = parent_entropy - child_entropy
        return information_gain

    def _split(self, X_column, split_thresh):
        left_idx = np.argwhere(X_column <= split_thresh).flatten()
        right_idx = np.argwhere(X_column > split_thresh).flatten()

        return left_idx, right_idx

    def _entropy(self, y):
        hist = np.bincount(y)
        ps = hist / len(y)
        return -np.sum([p * np.log(p) for p in ps if p>0])

    def _most_common_label(self, y):
        counter = Counter(y)
        value = counter.most_common(1)[0][0]
        return value

    def predict(self, X):
        return np.array([self._traverse_tree(x, self.root) for x in X])

    def _traverse_tree(self, x, node):
        if node.is_leaf_node():
            return node.value

        if x[node.feature] <= node.threshold:
            return self._traverse_tree(x, node.left)
        return self._traverse_tree(x, node.right)

```

DecisionTree کلاس:

این کلاس درخت تصمیم را پیاده‌سازی می‌کند.

ویژگی‌های این کلاس عبارتند از :

`min_samples_split`: حداقل تعداد نمونه‌ها برای انجام یک تقسیم

`max_depth`: حداکثر عمق ممکن برای درخت

`n_features`: تعداد ویژگی‌هایی که بررسی می‌شوند تا بهترین تقسیم را پیدا کنیم

`root`: ریشه‌ی درخت

`_grow_tree(X, y, depth)`: متد خصوصی برای ساخت درخت با استفاده از بازگشتی

`_best_split(X, y, feat_idx)`: بررسی تمامی تقسیم‌ها برای ویژگی‌ها و یافتن بهترین تقسیم بر اساس افزایش اطلاعات

`_information_gain(y, X_column, threshold)`: محاسبه افزایش اطلاعات بر اساس تقسیم فعلی

`_split(X_column, split_thresh)`: تقسیم داده‌ها بر اساس آستانه مشخص شده

`_entropy(y)`: محاسبه آنروپی

`_most_common_label(y)`: یافتن بیشترین تکرار برچسب در بردار  $y$

`predict(X)`: پیش‌بینی خروجی برای مجموعه‌ی ورودی ایکس

`_traverse_tree(x, node)`: اجرای پیمایش درخت برای یک نمونه ورودی و یک گره

## 2. نحوه ی کار :

در این بخش، یک مجموعه داده تعریف شده است که برخی از مشخصات مربوط به یک رستوران و خروجی مورد نظر (منتظر بمانید یا نه) را شامل می‌شود.

سپس متغیرهای دسته‌بندی شوند.

سپس داده‌ها به دو مجموعه آموزش و آزمون تقسیم شده و درخت تصمیم آموزش داده می‌شود.


در نهایت، پیش‌بینی‌ها بر روی داده‌های آزمون انجام شده و دقت مدل محاسبه می‌شود.

### 3. دقت:

دقت به این معناست که چه تعداد از نمونه‌های تست به درستی پیش‌بینی شده‌اند. برای محاسبه دقت، ابتدا باید نتایج پیش‌بینی شده را با نتایج واقعی مقایسه کرد.

Accuracy: 1.0

این به این معناست که مدل با دقت 100٪ درست پیش‌بینی کرده است.

روش هایی برای افزایش دقت : 

پیش‌پردازش داده‌ها:

گاهی اوقات انجام پیش‌پردازش روی داده‌ها (مثل استانداردسازی یا نرمال‌سازی) می‌تواند به مدل کمک کند تا بهتر عمل کند.

استفاده از انتخاب ویژگی‌ها:

ممکن است بعضی از ویژگی‌ها تاثیر بیشتری در پیش‌بینی داشته باشند. انتخاب ویژگی‌های مناسب می‌تواند کمک کند.

استفاده از یک الگوریتم متنوع‌تر:

Random Forest یا Gradient Boosting

رفع اورفیتینگ:

درخت‌های تصمیم به تمایل به یادگیری داده‌های آموزشی به خوبی، حتی تا حدی که به داده‌های نوین بر نمی‌خورند. این پدیده را اورفیتینگ می‌نامند. ممکن است نیاز باشد تا با تغییر پارامترها، تعداد نمونه‌های آموزشی یا استفاده از یک روش ارزیابی مختلف این موضوع را رفع کنیم.

توسعه یافتن مدل:

ممکن است با استفاده از مدل‌های پیچیده‌تر مانند شبکه‌های عصبی، دقت مدل بیشتر شود. اما در این صورت نیاز به مقدار بیشتری از داده‌های آموزشی و تنظیم پارامترها داریم.

انجام کراس والیدیشن:

اجرای کراس والیدیشن بر روی مدل می‌تواند بهترین تنظیمات پارامترها را بر اساس داده‌های آموزش و اعتبارسنجی به دست آورد.

## 4. Entropy and Gini Index :

هر دو معیار میزان ارتباط بین متغیرهای ورودی و متغیر خروجی را اندازه‌گیری می‌کنند.

آنتروپی:

معیاری است که نشان می‌دهد چقدر داده‌ها گسسته هستند. آنتروپی بالا به معنای نامطمئنی بیشتر است و آنتروپی پایین به معنای نامطمئنی کمتر است.

: Gini Index

یک معیار اندازه‌گیری خطای ممکن در پیش‌بینی است. هرچه قدر کمتر باشد مدل بهتر و پایدارتر است.