

**دانشکده‌ سامانه‌های هوشمند**

**یادگیری ماشین**

**استاد درس: دکتر سامان هراتی زاده**

**تمرین شماره 1**

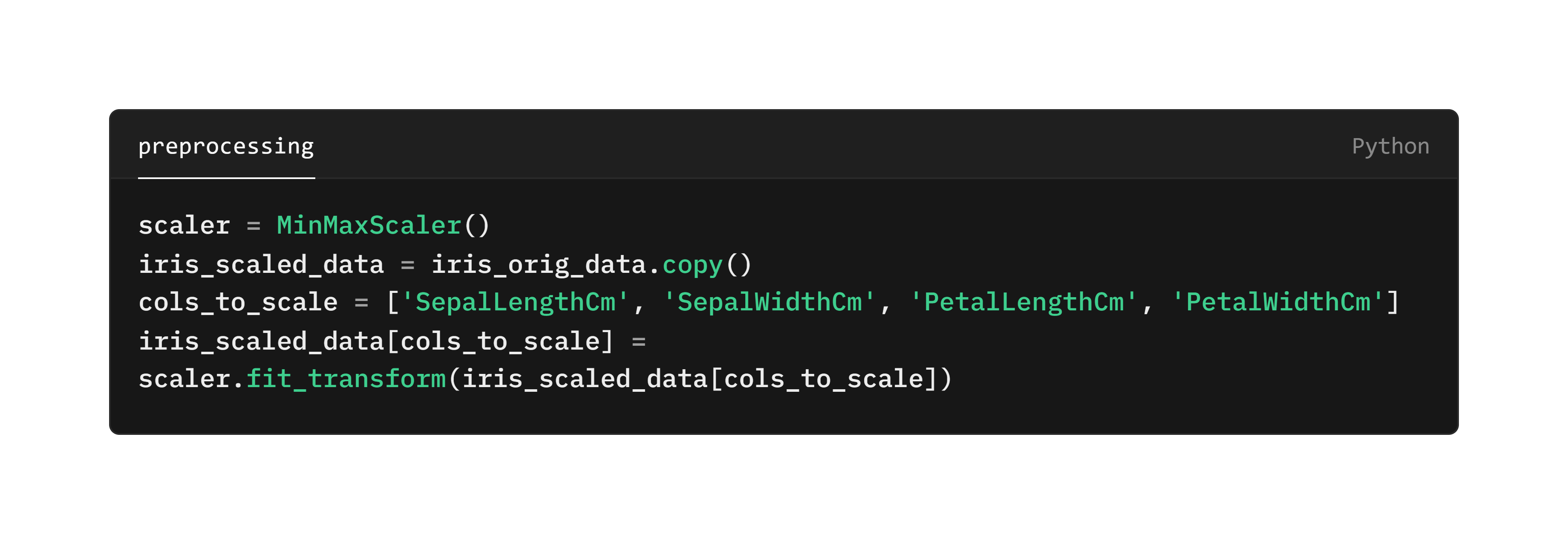
**تاریخ ثبت تمرین در سامانه: ...**

|  |  |
| --- | --- |
| محمدحسین مازندرانیان فرد | نام و نام خانوادگی |
| 830402066 | شماره‌ دانشجویی |
| 1 | شماره تمرین |
| 04/08/1403 | تاریخ ارسال گزارش |
| 10 ساعت | مدت زمان صرف شده برای پاسخ‌دهی به تمرین |

**سوال 1) پیاده سازی مدل KNN بر روی دیتاست iris**

پیش پردازش:

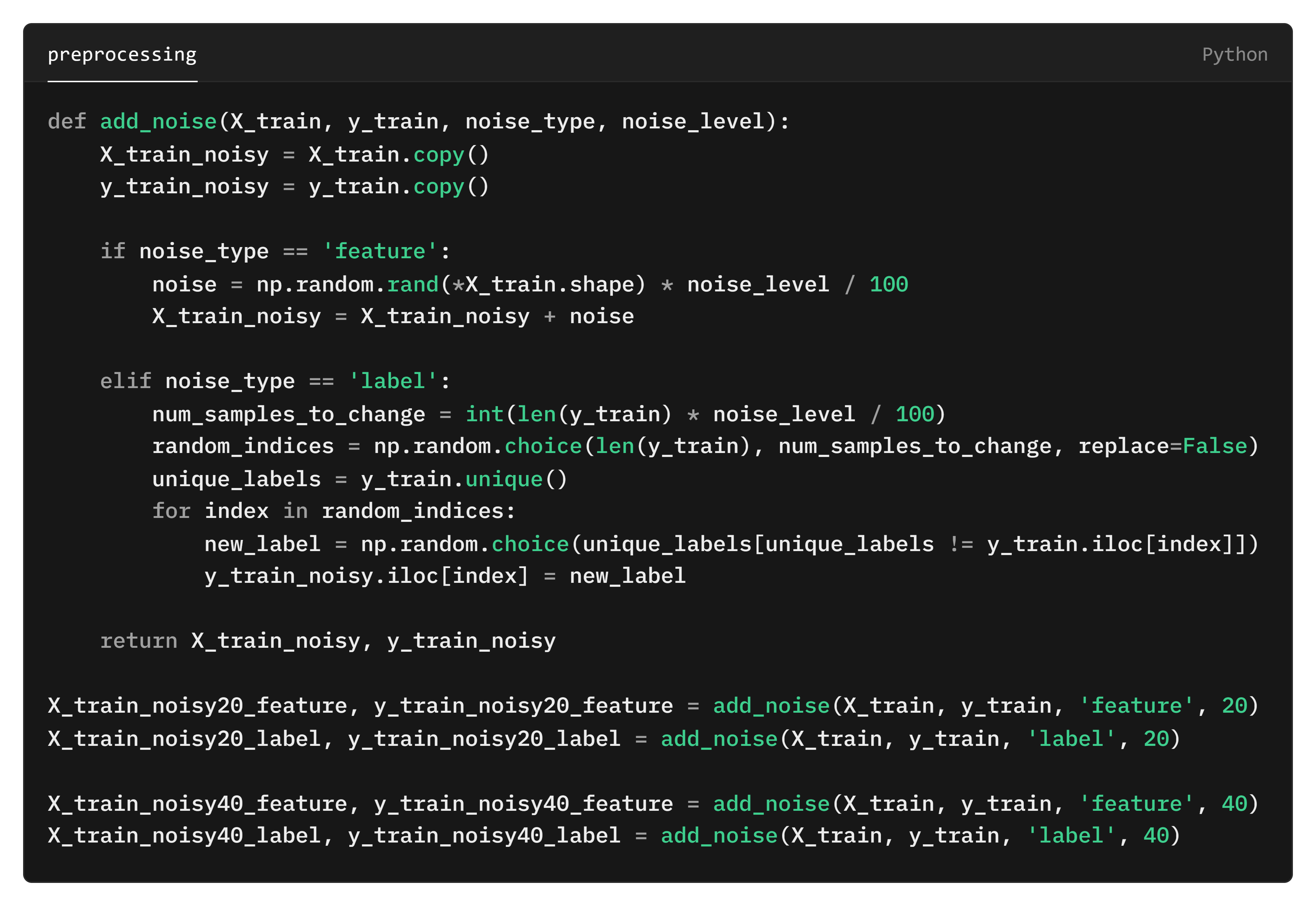
ابتدا ستون های عددی را با استفاده از روش min-max scaler بین بازه ی صفر تا یک نرمال میکنیم. دلیل اینکار این است.



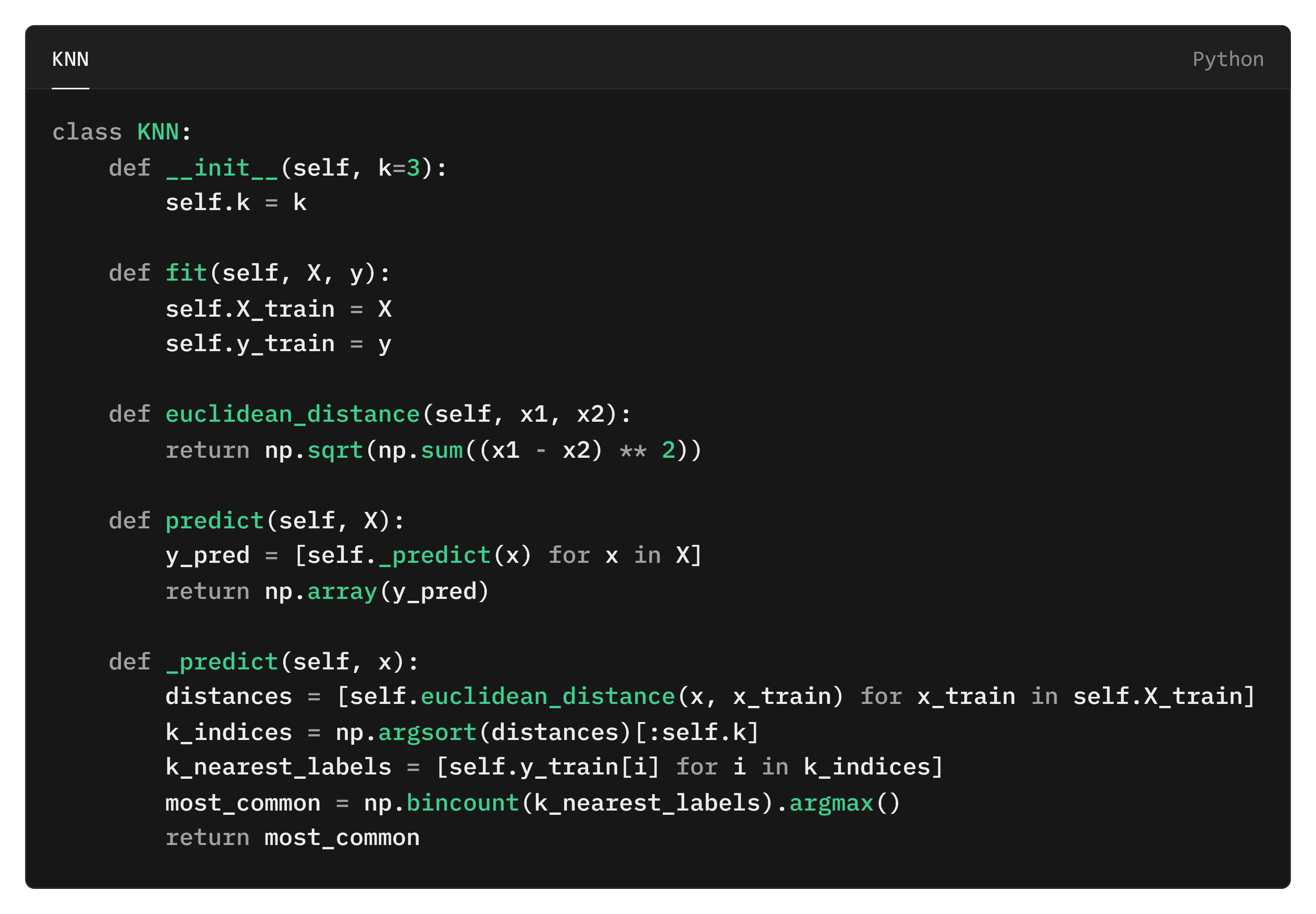
ستون Spices که ستون هدف ما هست را با استفاده از Label Encoder از داده های عددی به categorial تبدیل میکنیم:



در ادامه تابعی برای اضافه کردن نویز با مقادیر مختلف تعریف کردیم:



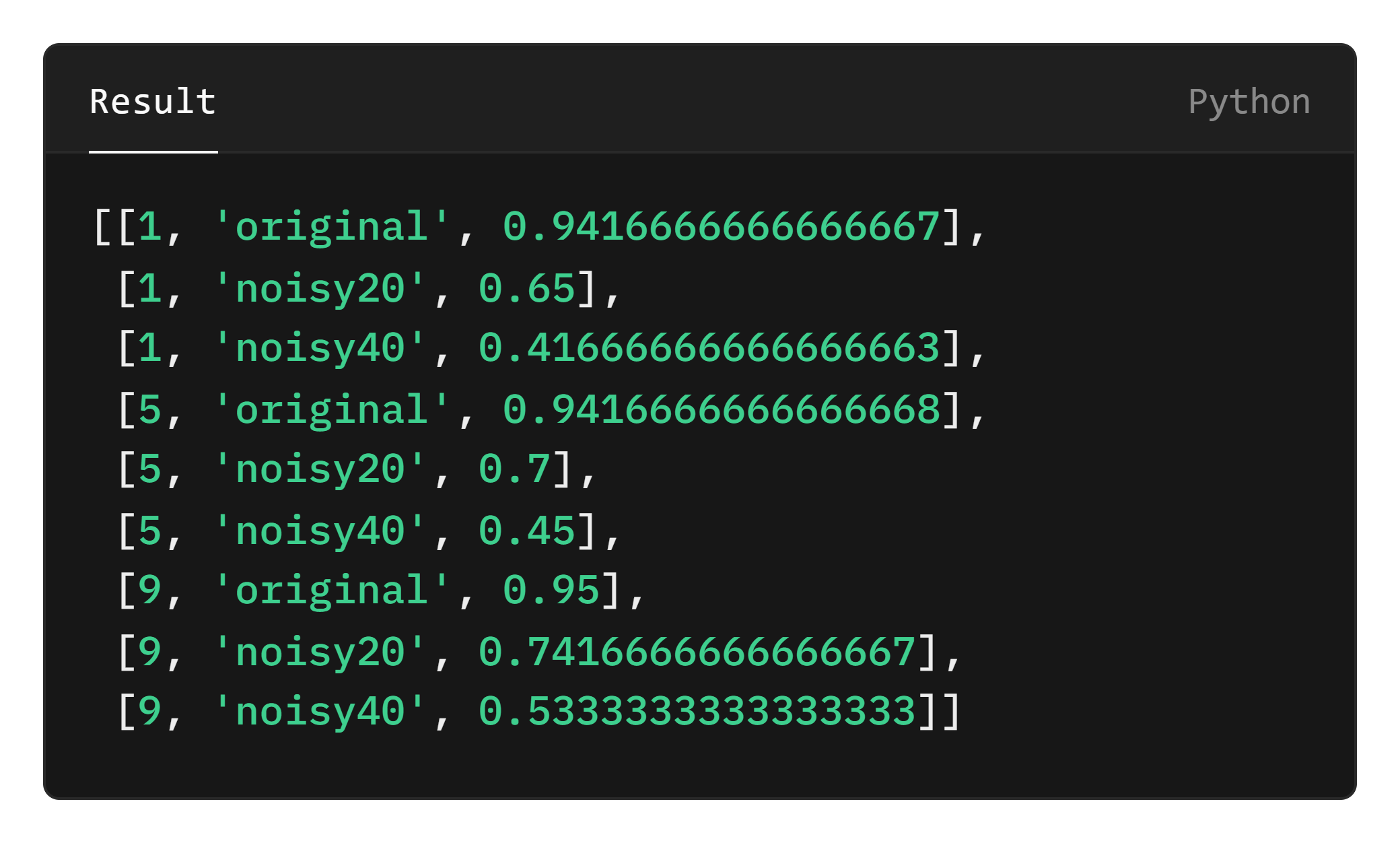
پیاده سازی مدل KNN:



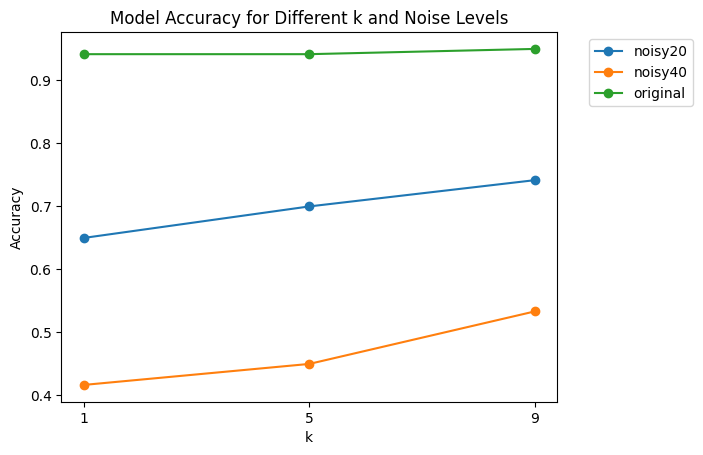
و در نهایت با استفاده از Five Fold Validation با مقادیر مختلف k مدل را تست می کنیم:



در نهایت دقت(accuracy) مدل روی دیتاهایی مختلف رو در زیر مشاهده می کنیم:

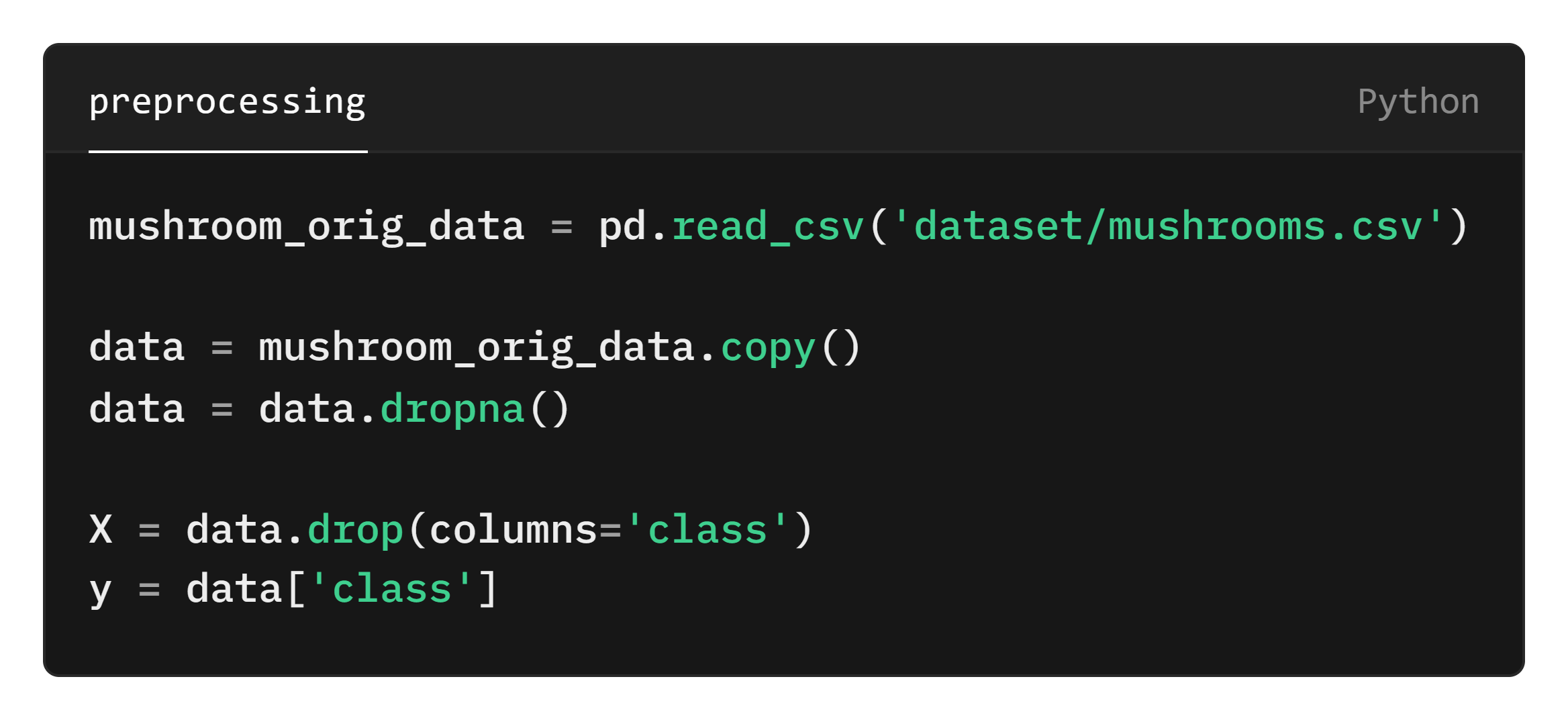


در حالت k = 9 و روی دیتای اورجینال بیشترین دقت بدست آمده است.



**سوال 2) پیاده سازی مدل Prism بر روی دیتاست mushrooms**

در پیش پردازش داده ها ردیف هایی که شامل missing value هستند را حذف کردیم، یک راه دیگر این است که missing value ها را به عنوان یک مقدار در نظر بگیریم.

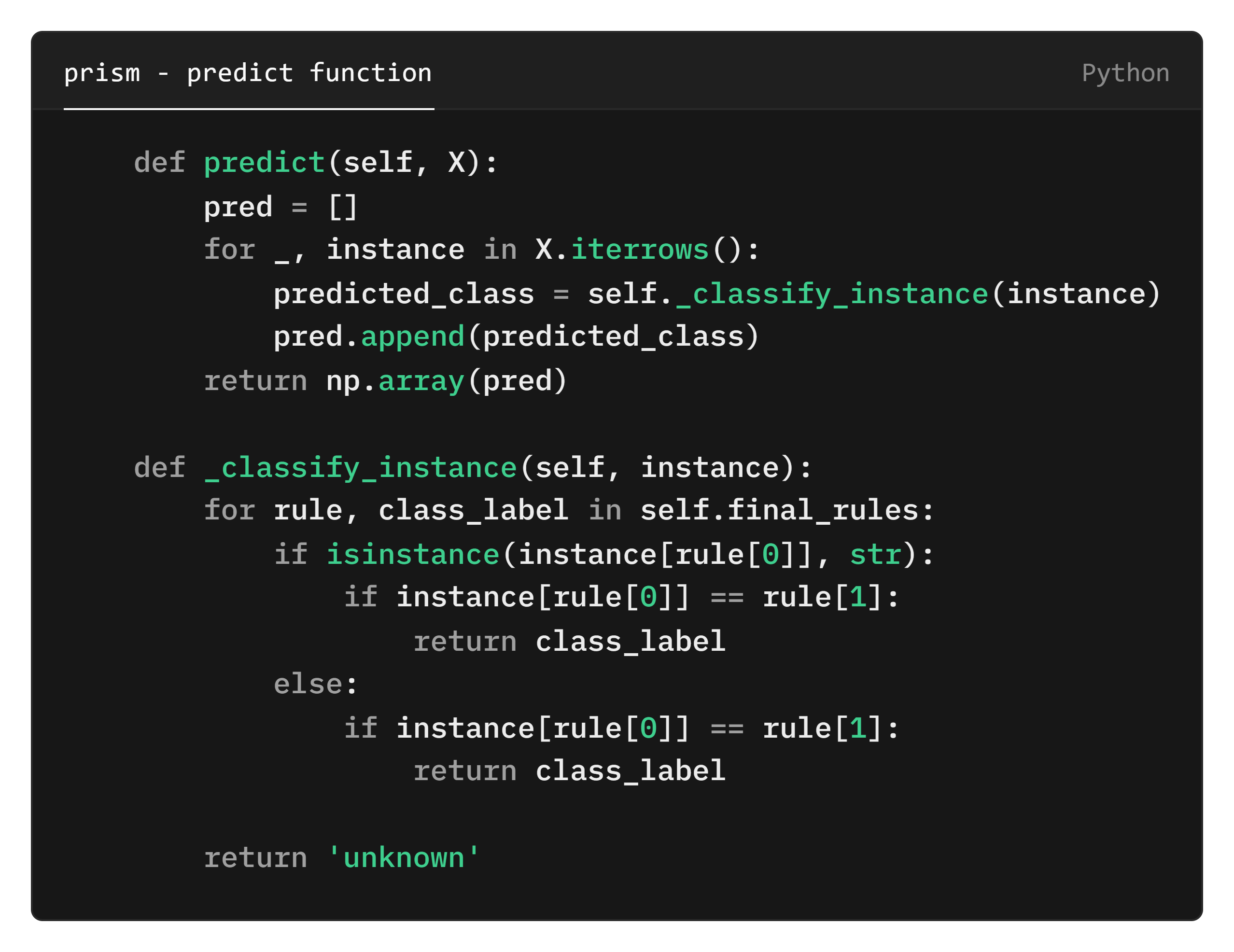


کلاس PrismClassifier شامل سه متد اصلی fit، predict و rules است که به طور خلاصه هر کدام را در ادامه توضیح می دهیم:

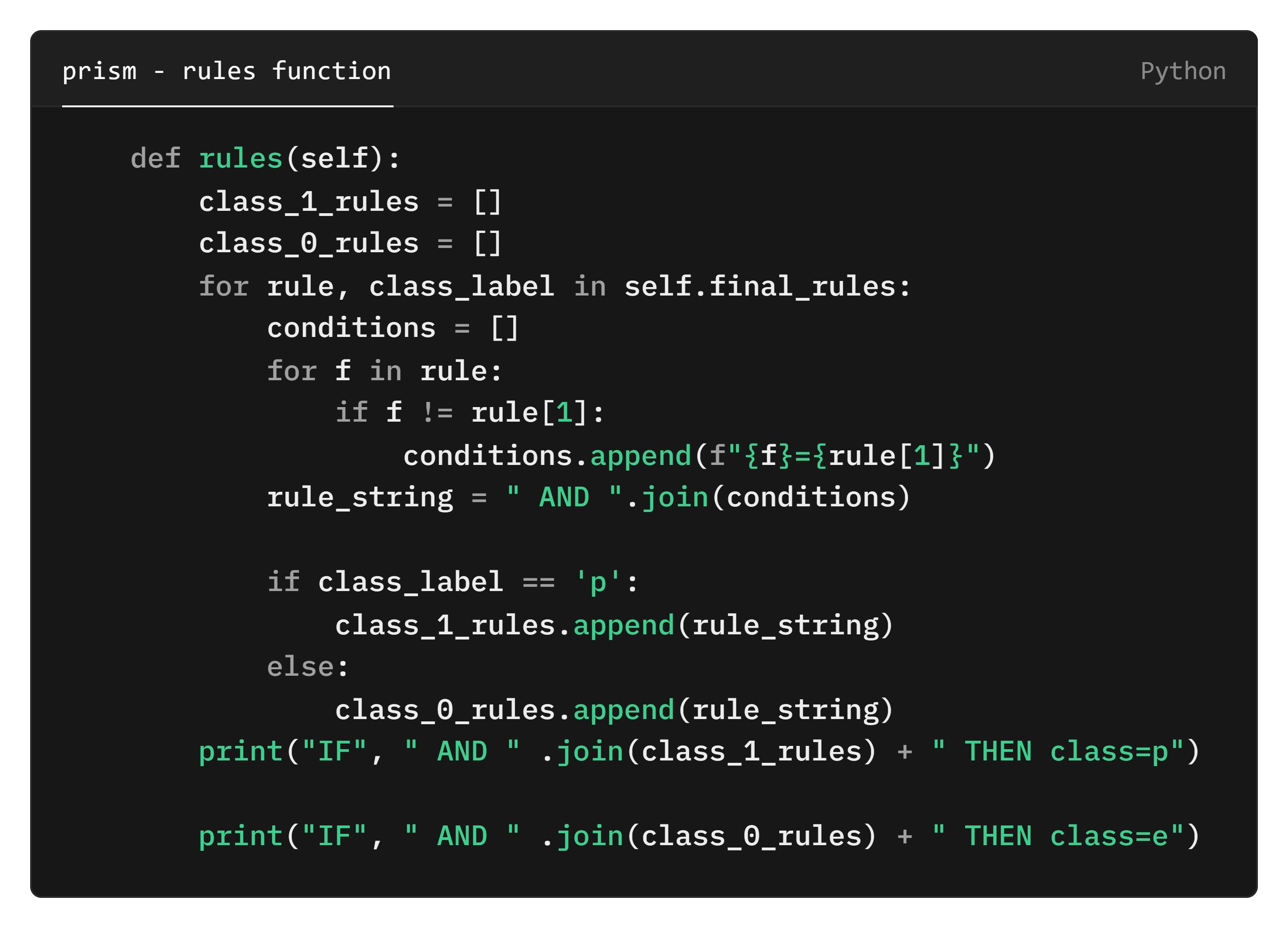
در متد fit ابتدا احتمال مقادیر هر feature مربوط به هر class را محاسبه می کنیم و سپس مثداری که بیشترین تکرار را دارد را انتخاب میکنیم. سپس ردیف هایی که این Rule پوشش می دهد را حذف میکنیم و دوباره روی ردیف های باقی مانده این عمل را تکرار می کنیم تا rules نهایی برای هر کلاس را پیدا کنیم.



در متد predict رو داده های هدف پیمایش انجام می دهیم و بر اساس final rules ای در مرحله ی fit پیدا کرده ایم برای هر ردیف مقدار ستون هدف را محاسبه می کنیم. در این مرحله اگر نتوانستیم مقداری پیدا کنیم unknown را انتخاب می کنیم.



در متد rules خروجی rules ها برای هر کلاس p و e را چاپ می کنیم:



نکته ای که در مرحله ی fit وجود دارد مقدار min\_support هست که rule هایی که کمتر از این مقدار حست را حساب نمی کنیم. بیشترین مقدار دقت این مدل برای وقتی بود که min\_support برابر 0.001 در نظر گرفته شد.

**سوال 3) پیاده سازی مدل ID3 بر روی دیتاست titanic**

برای پیش پردازش ستون fare را به چهار bin تقسیم کردیم و ستون های sex و embarked هم با استفاده از label encoder به داده های categorial تبدیل شدند:



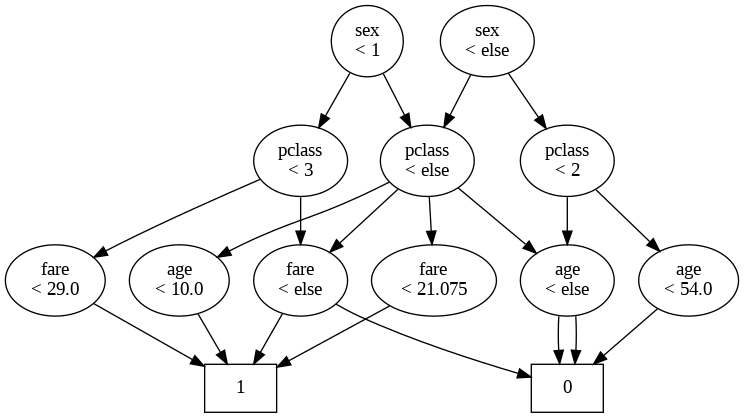
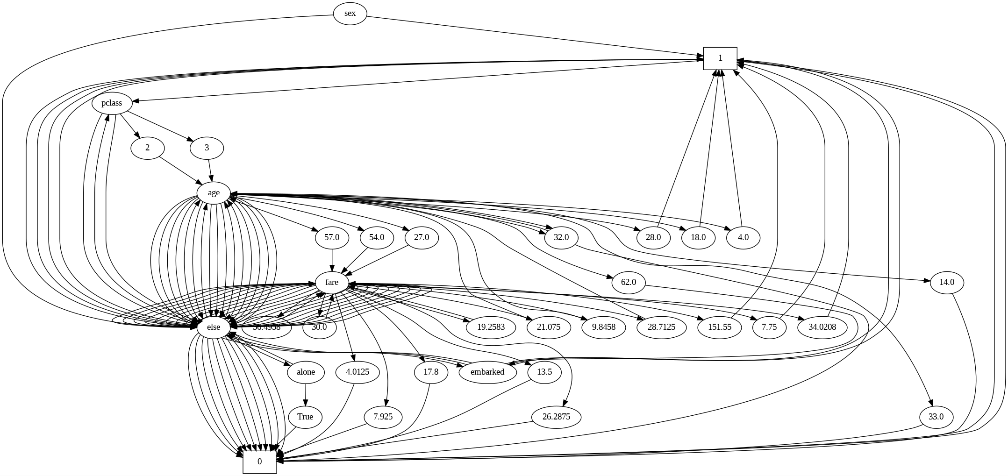
درخت را با استفاده از معیار های information gain و information gain و gini index و همچنین مقدار عمق درخت train کردیم و با استفاده از از دیتای ولیدیشن به بهترین پارامتر رسیدیم:



سپس با استفاده از همین معیار ها درخت را روی داده های تست predict کردیم که بهترین دقت به شرح زیر است:



نتایج رسم درخت با استفاده از کتابخانه ی graphviz:



**توضیح کلی**

* **کلاس ID3DecisionTree:**  ویژگی‌ها و معیارهای مختلف برای الگوریتم ID3 را به عنوان پارامتر می‌گیرد و سپس مدل را روی داده‌ها برازش داده و بر اساس ویژگی‌ها و آستانه‌های انتخاب‌شده یک درخت تصمیم‌گیری می‌سازد.
* **متدهایfit و predict:**  برای ساخت درخت تصمیم با داده‌های آموزشی و predict برای پیش‌بینی با داده‌های جدید استفاده می‌شوند.

**توضیح متد build\_tree**

متد build\_tree یکی از اجزای اصلی این کلاس است که ساختار درخت تصمیم را به صورت بازگشتی ایجاد می‌کند:

1. **پایان بازگشتی**: این تابع اگر به شرط‌های پایان بازگشتی برسد (یعنی عمق درخت به حداکثر max\_depth برسد، تمام نمونه‌ها از یک کلاس باشند یا تعداد نمونه‌ها کمتر از min\_samples\_split باشد)، یک گره برگ ایجاد می‌کند که دارای مقدار پرتکرارترین کلاس است.
2. **یافتن بهترین ویژگی و آستانه**: اگر به شرایط پایان نرسد، متد find\_best\_split را فراخوانی می‌کند تا بهترین ویژگی و آستانه‌ای که اطلاعات بیشتری را ارائه می‌دهد، پیدا کند. این انتخاب بهترین ویژگی و آستانه بستگی به معیار انتخاب‌شده مانند اطلاعات متقابل یا جینی دارد.
3. **تقسیم داده‌ها**: اگر ویژگی مناسبی یافت شد، نمونه‌ها را به دو گروه تقسیم می‌کند: یکی برای مقادیر کمتر از آستانه و دیگری برای مقادیر برابر یا بیشتر از آستانه.
4. **ساخت زیر درخت‌ها**: سپس این متد به صورت بازگشتی زیر درخت‌های سمت چپ و راست را می‌سازد و این تقسیمات را ادامه می‌دهد تا به یکی از شرایط پایان برسد.
5. **برگشت ساختار درخت**: پس از ساخت زیر درخت‌های چپ و راست، یک دیکشنری برمی‌گرداند که ویژگی انتخابی را با دو مقدار آستانه و "else" (برای شاخه دیگر) به عنوان زیر درخت‌ها نشان می‌دهد.

این ساختار درخت در نهایت به عنوان مدل ساخته‌شده در متغیر self.tree ذخیره می‌شود و برای پیش‌بینی روی داده‌های جدید استفاده می‌شود.

