```
from operator import xor
from django.db import models
from django.urls import reverse
from django.utils.translation import gettext_lazy as _
from django.core.exceptions import ValidationError

from oscar.core.utils import slugify
from oscar.apps.catalogue.abstract_models import AbstractProductAttribute, AbstractProduct
from oscar.models.fields import AutoSlugField


class ProductAttribute(AbstractProductAttribute):
    """
    Subclass of the AbstractProductAttribute class, adding metal and primary_gemstone fields.
    """
    metal = models.ForeignKey(
        'catalogue.Metal', on_delete=models.PROTECT, related_name='product_attribute',
        verbose_name='Metal Type', blank=True, null=True)

    primary_gemstone = models.ForeignKey(
        'catalogue.Gemstone', on_delete=models.PROTECT, related_name='product_attribute',
        verbose_name='Primary Gemstone Type', blank=True, null=True)

    common_attribute = models.BooleanField(verbose_name='Common Attribute (Y/N)')

    def get_absolute_url(self):
        return reverse('dashboard:attribute_detail', args=[str(self.id)])

    def clean(self):
        super().clean()

        if (
                (
                        self.common_attribute and
                 (self.metal or self.primary_gemstone)
                )
                or
                (
                        not self.common_attribute and
                 not xor(bool(self.metal), bool(self.primary_gemstone))
                )
        ):

                raise ValidationError(_("Product Attribute can belong to exactly one category: "
                                        "Metal, Gemstone or Common Attribute."))


class Product(AbstractProduct):
    """
    Subclass of the AbstractProduct class, adding metal and primary_gemstone fields.
    """
    metal = models.ForeignKey(
        'catalogue.Metal', on_delete=models.PROTECT, related_name='product',
        verbose_name='Metal Type', null=True)

    primary_gemstone = models.ForeignKey(
        'catalogue.Gemstone', on_delete=models.PROTECT, related_name='product',
        verbose_name='Primary Gemstone Type', null=True)

    def get_metal_type(self):
        """
        Return a product's metal type. Child products inherit their parent's.
        """
        if self.is_child:
            return self.parent.metal
        else:
            return self.metal
    get_metal_type.short_description = _('Metal type')

from oscar.apps.catalogue.models import *  # noqa isort:skip
```