Fairly Made technical test

Pierre MENNESSON

1 Project Description

The goal of this project is to build a predictive model to estimate the environmental impact of a fashion product given some key features such as its composition, manufacturing country, type of clothes etc... and then use this model to advise Fairly Made customers on reducing this impact at minimum cost.

2 Investigating the dataframe, features encoding

Each item comes with 10 (possibly unknown) features : *'product_type', 'nb_components', 'composition', 'raw_material_country', 'weaving_country', 'dyeing_country', 'manufacturing_country', 'plane_in_transports', 'climate_change', 'resource_use_fossils'*.

We would like to infer the *'climate_change'* and/or emph'resource_use_fossils' variables given the other ones, let's focus on the first one. After removing duplicates and entries with empty *'climate_change'* value, we're left with 7966 rows and 9 columns.

```
df=pd.read_csv('./fairlymade_products_impacts.csv')
df=df.dropna(subset=['climate_change'])
df=df.drop(columns=["study_id","resource_use_fossils"])
df=df.drop_duplicates()
df
```

	product_type	nb_components	composition	raw_material_country	weaving_country	dyeing_country	manufacturing_country	plane_in_transports	climate_change
0	ZIPPED COATS	2	POLYESTER, RECYCLED POLYAMIDE	CN	CN	CN	CN	False	2.111579
1	BODYSUITS	1	COTTON, ELASTANE	CN	NaN	PT	PT	False	4.360929
2	SHIRT-DRESSES	2	VEGETAL IVORY - COROZO, COTTON	NaN	PK	PK, CN	IT	False	12.763464
3	LEGGINGS	3	POLYESTER, ELASTANE, POLYAMIDE	NaN	CN	CN	PL	False	7.265380
4	ZIPPED COATS	3	RECYCLED BRASS, RECYCLED POLYESTER, ZAMAK, NYL...	IT	NaN	CN	HK	False	35.347939
...	...	...	...	...	...	...	...	...	...
8914	JUMPSUITS & BOILERS	2	VISCOSE, RECYCLED POLYESTER, VIRGIN WOOL, ELAS...	TR	IT	IT	BG	False	30.229603
8915	JUMPSUITS & BOILERS	1	ELASTANE, ORGANIC COTTON	NaN	TR	TN	TN	False	14.551640
8916	JUMPSUITS & BOILERS	1	ORGANIC COTTON	NaN	TR	TR	TR	False	12.803686
8917	JUMPSUITS & BOILERS	1	ORGANIC COTTON	NaN	TR	TR	TR	False	12.639963
8918	JUMPSUITS & BOILERS	1	ORGANIC COTTON	NaN	TR	TN	TN	False	8.103769

7966 rows × 9 columns

The different features (columns) come in different types :

(i) *categorical features* : features whose values belong to a finite set with no ordinal relationships, for instance *'product_type'* which belongs to a finite set of clothes (COATS, JEANS etc...).

(ii) *multi-valued categorical features* : features that take *multiple* values among a finite set or, in a more formal way, among the power set of a finite set. For instance, let's consider the set S of all materials used in the fashion industry. Then the compostion variable could be [ELASTANE], [ELASTANE,ORGANIC COTTON] or [SILK, POLYESTER, RECYCLED POLYAMIDE] which are subsets of S, i.e. elements of the power set $\mathcal{P}(S)$. .

The other ones are 'raw_material_country', 'weaving_country', 'dyeing_country', 'manufacturing_country' where any process can be done in several countries at the same time..

(iii) *numerical features* : features expressed as a real number like our target variable 'climate_change'.

(iv) *ordinal features* : just like categorical features but with an ordering relation. For instance 'nb_components' take its values in $[1, 2, 3]$ but we know that $1 < 2 < 3$, there is no natural order for the product_type like JEANS<COATS. We could also treat this variable like a numerical feature or a regular categorical feature. For simplicity, we will see it as a categorical one. There is no loss for our model capacity into doing so and, because of the small number of values it can take, the risk of overfitting is small..

(v) *boolean features* : a particular type of categorical features taking only two values : True or False like 'plane_in_transports'.

Before building a predictive model to infer the climate_change value from these features, we need to turn them into a "more mathematical" object, i.e. a vector : a list of real numbers that can be fed to a mathematical function that will output the desired value.

(i) *categorical features* : A first (naive) way to do so would be to sort the different values in alphabetical order and then assign to each category its position (starting at 0) in the sorted list. For instance 'BIBS & BRACES' would turn into 0, 'BIKINIS' into 1, 'BLAZERS & SUITS' into 2 etc...

The problem with this representation is the induced distance between the values : why would 'BIKINIS' be closer to 'BLAZERS' than 'ZIPPED COATS' ? Most of Machine Learning models are modelized by continuous function, in the sense that, close inputs will output close values. We need

to pay attention to the geometry of our representation when turning categorical variables into vectors.

A better (yet more expensive) representation is the *one hot encoding* : we have 34 possible values for the product_type feature so let's transform each possible value to a vector of size 34 with all coordinates equal to 0 except the one corresponding to its position in the sorted list equal to 1. Two distinct values are turned into distinct vectors (there is no loss of information) and all the vectors are equidistant in $\mathbb{R}^{34}$: the space of vectors of length 34.

We could actually take a 33 sized vector, encode the 34th value by the zero vector and the others values in the same way. We would still have a nice representation with (slightly) less memory usage. Still, we keep the 34 dimensions and reserve the zero vector for samples with missing product_type value.

(ii) *multi-valued categorical features* : Let's look at the composition variable. A component is a subset of a set with 117 elements. To encode a subset like ['SILK','COTTON'], we choose a vector of size 117 with all coordinates equal to zero except the ones corresponding to the position of 'SILK' and 'COTTON' in the sorted list of all possible components. Similarly, for the raw_material_country (resp. weaving_country, dyeing_country, manufacturing_country) we'll get a vector of length 35 (resp. 35,38,44).

(iii) *numerical features* : in our case, only the target variable climate_change is numerical but anyway : there is nothing to do.

(iv) *ordinal features* : we treat our unique ordinal feature nb_components just like a categorical feature and encode it with a vector of length 3.

(v) *boolean features* : it is quite simple : False $\to$ 0 and True $\to$ 1. We're treating the plane_in_transports variable like a categorical one at the difference that we are are actually dropping one coordinate. We can do so because there is no sample with missing plane_in_transports value and we obtain a number i.e. a vector of length 1.

By concatenating all these 0/1 vectors, we get a vector of length 34+117+35+35+38+44+3+1=307 that is ready to be fed to any standard Machine Learning predictor !

Remarks. *This representation is very* sparse*, i.e. our encoded samples lie in a very "small part" of* $\mathbb{R}^{307}$ *and training a predictor on this large space may quickly overfit.*

— *Each coordinate can take up to only 2 values 0 or 1. This won't be a problem because our model (decision tree introduced shortly after) does a*

really good job with binary vectors : each binary coordinate will appear at most once when building the tree.

— *Even when seen as a subset of* $\{0,1\}^{307}$*, our samples (as well as the unobserved ones) will occupy a really thin part of this set. Some configurations are strictly forbidden like several 1s for a categorical variable (a sample can not be a COAT and a JEAN at the same time) and some are really unlikely like a lot of 1s for a multi categorical feature (a sample that has been manufactured in 30 different countries).*

Again, this won't be a problem for decision trees as their depth will scale to the size of the dataset instead of the representation space.

3 Model Selection : Congratulations to Random Forests

In our case, we have $m = 7965$ samples that are encoded to vectors of length $n = 307$ that we will denote $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ and we will write $y_1, \ldots, y_m$ their corresponding target climate_change value.

A model is a function $F : \mathbb{R}^n \to \mathbb{R}$. Finding the most accurate one is equivalent to find the one that minimizes the *mean squared loss* $L(F) = \frac{1}{m}\sum_{i=1}^{m}(F(x_i) - y_i)^2$.

We could test a given model on the whole dataset but we won't know the validity of our model on unseen data. Some models might do an excellent job on the available data but behave poorly on new samples. It's similar to test a math student by giving him exercises already seen as examples in the lecture : he could have a brain big enough to learn by heart all the answers yet not understanding the true mechanism.

This is why we keep some data apart, the *validation set* and optimize our model parameters on the remaining data the *training set* and select our final model according to the loss on the validation set.

After splitting our data into a train/validation sets, we try different standard Machine Learning Models : Ridge Regression, Support Vector Regressor, Decision Tree Regressors and Random Forests. It turns out that Random Forests seem to be the best predictor. As their name suggests, Random Forests combine several more humble predictors known as Decision Tree, let's take at look at how they work.

4 Decision Trees

We will focus on decision trees because
(i) They are the cornerstone of the model selection contest winners models : random forests.
(ii) They can handle our new binary features very well.
(iii) They are highly interpretable which will help us advise Fairly Made customers to reduce their environmental impact.

Before dwelling on the topic, let's see what could the most simple predictor possible ? What if F was indeed constant equal to c, i.e. the response would not even depend on the input features ?

To find the best c, let's minimize $\frac{1}{m}\sum_{i=1}^{m}(c-y_i)^2$ seen as a function of c. After derivating this function, we find that the best c is the mean of the samples target value $\hat{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$ and the minimum value at this point equals to the variance of the target values $\frac{1}{m}\sum_{i=1}^{m}(\hat{y}-y_i)^2$.

Obviously this is a poor model, but let's see what happens if we split our data in two subsets, for instance by choosing a binary coordinate and dividing the samples according to the coordinate value (0 or 1).

Let I_0 (resp. I_1) be the indexes of samples having the coordinate equal to 0 (resp. 1) and let's assign the value $\hat{y}_0 = \frac{1}{|I_0|}\sum_{i\in I_0} y_i$ (resp. $\hat{y}_1 = \frac{1}{|I_1|}\sum_{i\in I_1} y_i$) where $|I_0|$ (resp. $|I_1|$) denotes the number of elements in I_0 (resp. I_1).

The new loss equals to

$$\frac{1}{m}\sum_{i\in I_0}(y_i-\hat{y}_0)^2+\frac{1}{m}\sum_{i\in I_1}(y_i-\hat{y}_1)^2 = \frac{|I_0|}{m}\frac{1}{|I_0|}\sum_{i\in I_0}(y_i-\hat{y}_0)^2+\frac{|I_1|}{m}\frac{1}{|I_1|}\sum_{i\in I_1}(y_i-\hat{y}_1)^2$$

.

Or, we now that $\frac{1}{|I_0|}\sum_{i\in I_0}(x_i-\hat{y}_0)^2 \leq \frac{1}{|I_0|}\sum_{i\in I_0}(x_i-\hat{y})^2$ and

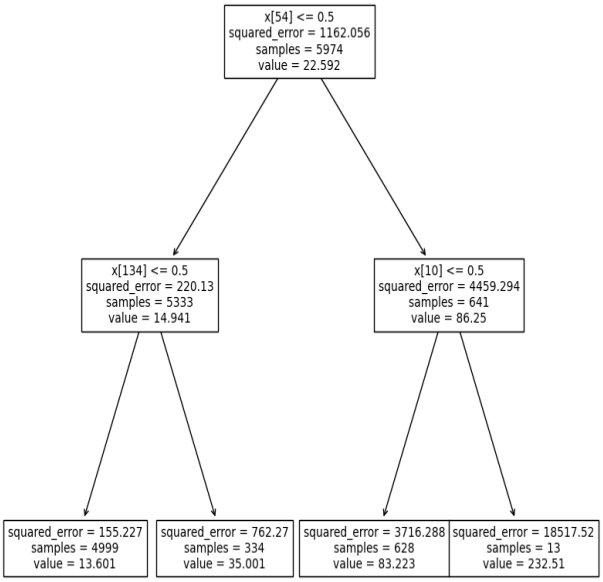
$\frac{1}{|I_1|}\sum_{i\in I_1}(y_i-\hat{y}_1)^2 \leq \frac{1}{|I_1|}\sum_{i\in I_1}(y_i-\hat{y})^2$ because of the squared error minimizing property of the mean. The new loss is less or equal to

$$\frac{|I_0|}{m}\frac{1}{|I_0|}\sum_{i\in I_0}(y_i-\hat{y})^2+\frac{|I_1|}{m}\frac{1}{|I_1|}\sum_{i\in I_1}(y_i-\hat{y})^2 = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}-y_i)^2$$

which is the old loss.

In other words, splitting our dataset into two groups and assigning to each group its mean target value always reduces the loss and the new loss is the weighted mean of the subsets losses. Repeating this operation for each subset recursively, by choosing the coordinate that reduces the variance (= the local loss minimum) the most, yields a tree-like structure.

In the figure below, we first split our dataset according to the 54th coordinate, then split the first subset according to the 134th coordinate and the second according to the 10th coordinate.



When this process will stop ? If, within all the children subsets, samples share common values so that there is no way to split the subset again along a coordinate.

We may introduce some stopping criterion to limit the depth of the tree and avoid this situation that is likely to give birth to an overfitted model.

(i) *max_depth* : the maximum depth of the tree.

(ii) *min_samples_leaf* : the minimum number of samples a leaf (extreme node) should have. A node will be split if and only if both its children contain at least min_samples_leaf samples.

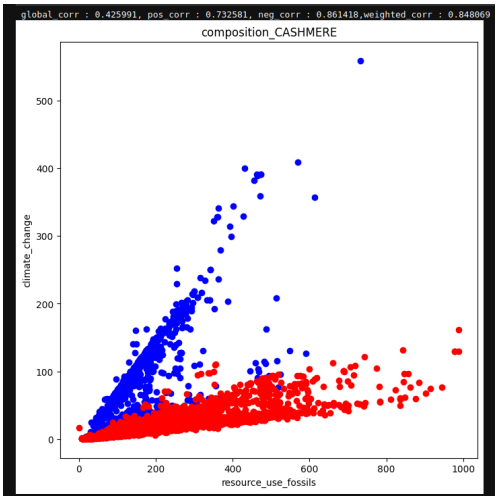
(iii) *min_impurity_decrease* : the minimum gap between the old target value variance of a node and the weighted variances of its children i.e. a lower bound when decreasing the loss.

Once the tree is built, when observing a new sample during inference, we simply follow the path in the tree by looking at the relevant coordinates until we reach a node that won't be split again : *a leaf*. The output value is the mean value of the training samples contained in this leaf.

5 Bonus : automatic feature selection for better data visualization insights

As said before, decision trees are highly interpretables (in contrast to neural networks for instance). In particular, the sklearn DecisionTreeRegressor object tells us, after training, which features have the biggest impact in our prediction.

We can leverage this information to choose how we want to visualize our samples. In our case, the most important binary feature is "composition_CASHMERE", i.e. whether a product contains cashmere or not. Let's plot the samples climate_change and resource_use_fossils samples values in two dimensions and color the samples according to whether or not they contain cashmere :



We observe see that the presence of cashmere in a fashion item has some influence on the correlation between the variables climate_change and resource_use_fossils : whether or not cashmere appears in the composition (blue points) or not (red points) the resource_use_fossils and the climate_change variable look linearly correlated but with different coefficients.

We can also compute the Pearson correlation coefficient to verify that, conditioning the variable on the presence of cashmere yields a stronger linear relationship (see the print).

6 Reducing the environmental impact

Given a fashion item, how can we use our decision tree to reduce its ecological impact at minimum cost ? During inference, the sample will end up in a leaf and we must look at what features we may modify for it to end up in a leaf

with a lower value.

Each leaf correspond to a partial coordinates assignment to 0 or 1 which can be seen as a bunch of conditions on each feature :

(i) For a categorical feature, it corresponds to either several forbidden values or to one mandatory value. For instance, for the product_type, it could either mean "not a element of {JEANS, FORMAL COATS, UNDERWEARS}" or "must be equal to CARDIGANS".

(ii) For a multi-valued categorical feature, it corresponds to some forbidden and some mandatory values. For instance, for the composition, it could mean "does not contain SILK or COTTON but contains POLYESTER".

Knowing the expenses of removing one component or relocating some process from a country to another provides us with a cost function between a fashion item and a configuration leaf. For instance, the cost of modifying this sample

	product_type	nb_components	composition	raw_material_country	weaving_country	dyeing_country	manufacturing_country	plane_in_transports
0	SHIRTS	2	FLAX, CASHMERE	CN	FR	FR, JP	PL, FR	True

to meet one leaf conditions such as

```
{'product_type': {'label': 'SHIRTS'},
 'nb_components': {'forbidden_labels': array([], dtype=int64)},
 'composition': {'forbidden_labels': array(['ALPACAWOOL', 'BRASS', 'CASHMERE', 'MERINOWOOL', 'ORGANICCOTTON',
        'POLYAMIDE', 'POLYESTER', 'SHEEPWOOL', 'VIRGINWOOL', 'VISCOSE',
        'ZAMAK'], dtype='<U33'),
  'mandatory_labels': array(['COTTON', 'FLAX'], dtype='<U33')},
 'raw_material_country': {'forbidden_labels': array([], dtype='<U2'),
  'mandatory_labels': array([], dtype='<U2')},
 'weaving_country': {'forbidden_labels': array(['CN', 'TN'], dtype='<U2'),
  'mandatory_labels': array([], dtype='<U2')},
 'dyeing_country': {'forbidden_labels': array(['BD', 'CN', 'JP'], dtype='<U2'),
  'mandatory_labels': array([], dtype='<U2')},
 'manufacturing_country': {'forbidden_labels': array(['FR', 'IN', 'PK', 'TR', 'VN'], dtype='<U2'),
  'mandatory_labels': array(['IT'], dtype='<U2')},
 'plane_in_transports': True}
```

amounts to the sum of the following costs :

(i) Replace the CASHMERE component by a component in COTTON.

(ii) Relocate the dyeing process occurring in JP in any other country except BN and CN.

(iii) Relocate the manufacturing process occurring in FR to IT.

The cost function should not be hard to code (granted the knowledge of the actual financial costs). Now, given a fashion item and a maximum budget B we can

(a) Compute the sample cost from all the possible leaf configurations (in our case, the tree has 558 leaves which can be computed quickly).

(b) Select the configurations where the computed costs is below B

(c) Among these configurations, choose the one with the lower climate_change value

Remarks. *Obviously, the customer would like to keep producing the same kind of article. To get rid of useless leaf configurations, we can define our cost function to be $+\infty$ if the sample product_type differs from the label value of the configuration or belongs to the forbidden values.*

7 Further Directions

(A) Gather the necessary data in order to design the cost function.
(B) The best model we trained so far were Random Forests which combine the individual predictions of trees trained by randomly sampling samples and features. We can still adapt our method to identify leaf configurations and compute the cost from a new sample to all the leaves.
(C) Try other ensemble methods.
(D) Try to incorporate our knowledge directly into the model : producing in China is a priori worse than producing in Italy. This could be used to build the first branches of the tree and then let the data build the next ones.
(E) The author did not guess whether the resource_use_fossils should be considered as an input feature (a known feature provided by a customer) or some target variable to be inferred (like climate_change).

Trying to predict resource_use_fossils directly from the same binary features gave worse scores than for the climate_change value. Nevertheless, including resource_use_fossils as input feature really improved our predictions for the climate_change value.

This was done using a mix of Decision Trees and Ridge Regression (see https://pypi.org/project/linear-tree/) and gave way better results (see LMT.ipynb notebook) with a R^2 score ≈ 0.65 when training a Decision Tree only on categorical features versus ≈ 0.85incorporating the resource_use_fossils in a Linear Model Tree.