NBA Players Succes Estimation Perform Feature Engineering

May 9, 2025

1 Perform feature engineering

1.1 Introduction

As you're learning, data professionals working on modeling projects use featuring engineering to help them determine which attributes in the data can best predict certain measures.

In this activity, you are working for a firm that provides insights to the National Basketball Association (NBA), a professional North American basketball league. You will help NBA managers and coaches identify which players are most likely to thrive in the high-pressure environment of professional basketball and help the team be successful over time.

To do this, you will analyze a subset of data that contains information about NBA players and their performance records. You will conduct feature engineering to determine which features will most effectively predict whether a player's NBA career will last at least five years. The insights gained then will be used in the next stage of the project: building the predictive model.

1.2 Step 1: Imports

Start by importing `pandas`.

[1]:
```
# Import pandas.

### YOUR CODE HERE ###

import pandas as pd
```

The dataset is a .csv file named `nba-players.csv`. It consists of performance records for a subset of NBA players. As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[2]:
```
# RUN THIS CELL TO IMPORT YOUR DATA.

# Save in a variable named `data`.

### YOUR CODE HERE ###

```

```
data = pd.read_csv("nba-players.csv", index_col=0)
```

Hint 1

The `read_csv()` function from `pandas` allows you to read in data from a csv file and load it into a DataFrame.

Hint 2

Call the `read_csv()`, pass in the name of the csv file as a string, followed by `index_col=0` to use the first column from the csv as the index in the DataFrame.

1.3 Step 2: Data exploration

Display the first 10 rows of the data to get a sense of what it entails.

[3]:
```
# Display first 10 rows of data.

### YOUR CODE HERE ###

data.head(10)
```

[3]:
```
              name  gp   min  pts  fgm  fga    fg  3p_made  3pa    3p  ...  \
0   Brandon Ingram  36  27.4  7.4  2.6  7.6  34.7      0.5  2.1  25.0  ...
1  Andrew Harrison  35  26.9  7.2  2.0  6.7  29.6      0.7  2.8  23.5  ...
2   JaKarr Sampson  74  15.3  5.2  2.0  4.7  42.2      0.4  1.7  24.4  ...
3      Malik Sealy  58  11.6  5.7  2.3  5.5  42.6      0.1  0.5  22.6  ...
4      Matt Geiger  48  11.5  4.5  1.6  3.0  52.4      0.0  0.1   0.0  ...
5     Tony Bennett  75  11.4  3.7  1.5  3.5  42.3      0.3  1.1  32.5  ...
6      Don MacLean  62  10.9  6.6  2.5  5.8  43.5      0.0  0.1  50.0  ...
7     Tracy Murray  48  10.3  5.7  2.3  5.4  41.5      0.4  1.5  30.0  ...
8     Duane Cooper  65   9.9  2.4  1.0  2.4  39.2      0.1  0.5  23.3  ...
9     Dave Johnson  42   8.5  3.7  1.4  3.5  38.3      0.1  0.3  21.4  ...

   fta    ft  oreb  dreb  reb  ast  stl  blk  tov  target_5yrs
0  2.3  69.9   0.7   3.4  4.1  1.9  0.4  0.4  1.3            0
1  3.4  76.5   0.5   2.0  2.4  3.7  1.1  0.5  1.6            0
2  1.3  67.0   0.5   1.7  2.2  1.0  0.5  0.3  1.0            0
3  1.3  68.9   1.0   0.9  1.9  0.8  0.6  0.1  1.0            1
4  1.9  67.4   1.0   1.5  2.5  0.3  0.3  0.4  0.8            1
5  0.5  73.2   0.2   0.7  0.8  1.8  0.4  0.0  0.7            0
6  1.8  81.1   0.5   1.4  2.0  0.6  0.2  0.1  0.7            1
7  0.8  87.5   0.8   0.9  1.7  0.2  0.2  0.1  0.7            1
8  0.5  71.4   0.2   0.6  0.8  2.3  0.3  0.0  1.1            0
9  1.4  67.8   0.4   0.7  1.1  0.3  0.2  0.0  0.7            0

[10 rows x 21 columns]
```

Hint 1

There is a function in the `pandas` library that can be called on a DataFrame to display the first n number of rows, where n is a number of your choice.

Hint 2

Call the `head()` function and pass in 10.

Display the number of rows and the number of columns to get a sense of how much data is available to you.

```
[4]: # Display number of rows, number of columns.

     ### YOUR CODE HERE ###

     data.shape
```

```
[4]: (1340, 21)
```

Hint 1

DataFrames in `pandas` have an attribute that can be called to get the number of rows and columns as a tuple.

Hint 2

You can call the `shape` attribute.

Question: What do you observe about the number of rows and the number of columns in the data? - There are 1,340 rows and 21 columns in the data.

Now, display all column names to get a sense of the kinds of metadata available about each player. Use the columns property in pandas.

```
[5]: # Display all column names.

     ### YOUR CODE HERE ###

     data.columns
```

```
[5]: Index(['name', 'gp', 'min', 'pts', 'fgm', 'fga', 'fg', '3p_made', '3pa', '3p',
            'ftm', 'fta', 'ft', 'oreb', 'dreb', 'reb', 'ast', 'stl', 'blk', 'tov',
            'target_5yrs'],
           dtype='object')
```

The following table provides a description of the data in each column. This metadata comes from the data source, which is listed in the references section of this lab.

Column Name	Column Description
`name`	Name of NBA player
`gp`	Number of games played
`min`	Number of minutes played per game
`pts`	Average number of points per game

Column Name	Column Description
`fgm`	Average number of field goals made per game
`fga`	Average number of field goal attempts per game
`fg`	Average percent of field goals made per game
`3p_made`	Average number of three-point field goals made per game
`3pa`	Average number of three-point field goal attempts per game
`3p`	Average percent of three-point field goals made per game
`ftm`	Average number of free throws made per game
`fta`	Average number of free throw attempts per game
`ft`	Average percent of free throws made per game
`oreb`	Average number of offensive rebounds per game
`dreb`	Average number of defensive rebounds per game
`reb`	Average number of rebounds per game
`ast`	Average number of assists per game
`stl`	Average number of steals per game
`blk`	Average number of blocks per game
`tov`	Average number of turnovers per game
`target_5yrs`	1 if career duration >= 5 yrs, 0 otherwise

Next, display a summary of the data to get additional information about the DataFrame, including the types of data in the columns.

[6]:
```python
# Use .info() to display a summary of the DataFrame.

### YOUR CODE HERE ###

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1340 entries, 0 to 1339
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         1340 non-null   object
 1   gp           1340 non-null   int64
 2   min          1340 non-null   float64
 3   pts          1340 non-null   float64
 4   fgm          1340 non-null   float64
 5   fga          1340 non-null   float64
 6   fg           1340 non-null   float64
 7   3p_made      1340 non-null   float64
 8   3pa          1340 non-null   float64
 9   3p           1340 non-null   float64
 10  ftm          1340 non-null   float64
 11  fta          1340 non-null   float64
 12  ft           1340 non-null   float64
 13  oreb         1340 non-null   float64
 14  dreb         1340 non-null   float64
 15  reb          1340 non-null   float64
 16  ast          1340 non-null   float64
 17  stl          1340 non-null   float64
 18  blk          1340 non-null   float64
 19  tov          1340 non-null   float64
 20  target_5yrs  1340 non-null   int64
dtypes: float64(18), int64(2), object(1)
memory usage: 230.3+ KB
```

Question: Based on the preceding tables, which columns are numerical and which columns are categorical? - Based on the preceding tables, the `name` column is categorical, and the rest of the columns are numerical.

1.3.1 Check for missing values

Now, review the data to determine whether it contains any missing values. Begin by displaying the number of missing values in each column. After that, use isna() to check whether each value in the data is missing. Finally, use sum() to aggregate the number of missing values per column.

[7]:
```
# Display the number of missing values in each column.
# Check whether each value is missing.
#Aggregate the number of missing values per column.

### YOUR CODE HERE ###

data.isna().sum()
```

[7]:
```
name           0
gp             0
min            0
pts            0
fgm            0
fga            0
fg             0
3p_made        0
3pa            0
3p             0
ftm            0
fta            0
ft             0
oreb           0
dreb           0
reb            0
ast            0
stl            0
blk            0
```

```
tov             0
target_5yrs     0
dtype: int64
```

Question: What do you observe about the missing values in the columns? All columns in this dataset have 0 missing values.

Question: Why is it important to check for missing values? Checking for missing values is an important step in data exploration. Missing values are not particularly useful, so it's important to handle them by cleaning the data.

1.4 Step 3: Statistical tests

Next, use a statistical technique to check the class balance in the data. To understand how balanced the dataset is in terms of class, display the percentage of values that belong to each class in the target column. In this context, class 1 indicates an NBA career duration of at least five years, while class 0 indicates an NBA career duration of less than five years.

[8]:
```
# Display percentage (%) of values for each class (1, 0) represented in the␣
↪target column of this dataset.

### YOUR CODE HERE ###

data["target_5yrs"].value_counts(normalize=True)*100
```

[8]:
```
1    62.014925
0    37.985075
Name: target_5yrs, dtype: float64
```

Hint 1

In pandas, `value_counts(normalize=True)` can be used to calculate the frequency of each distinct value in a specific column of a DataFrame.

Hint 2

After `value_counts(normalize=True)`, multipling by `100` converts the frequencies into percentages (%).

Question: What do you observe about the class balance in the target column? - About 62% of the values in the target columm belong to class 1, and about 38% of the values belong to class 0. In other words, about 62% of players represented by this data have an NBA career duration of at least five years, and about 38% do not. - The dataset is not perfectly balanced, but an exact 50-50 split is a rare occurance in datasets, and a 62-38 split is not too imbalanced. However, if the majority class made up 90% or more of the dataset, then that would be of concern, and it would be prudent to address that issue through techniques like upsampling and downsampling.

Question: Why is it important to check class balance?

If there is a lot more representation of one class than another, then the model may be biased toward the majority class. When this happens, the predictions may be inaccurate.

1.5 Step 4: Results and evaluation

Now, perform feature engineering, with the goal of identifying and creating features that will serve as useful predictors for the target variable, `target_5yrs`.

1.5.1 Feature Selection

The following table contains descriptions of the data in each column:

Column Name	Column Description
`name`	Name of NBA player
`gp`	Number of games played
`min`	Number of minutes played
`pts`	Average number of points per game
`fgm`	Average number of field goals made per game
`fga`	Average number of field goal attempts per game
`fg`	Average percent of field goals made per game
`3p_made`	Average number of three-point field goals made per game
`3pa`	Average number of three-point field goal attempts per game
`3p`	Average percent of three-point field goals made per game
`ftm`	Average number of free throws made per game
`fta`	Average number of free throw attempts per game
`ft`	Average percent of free throws made per game
`oreb`	Average number of offensive rebounds per game
`dreb`	Average number of defensive rebounds per game
`reb`	Average number of rebounds per game
`ast`	Average number of assists per game
`stl`	Average number of steals per game
`blk`	Average number of blocks per game
`tov`	Average number of turnovers per game
`target_5yrs`	1 if career duration >= 5 yrs, 0 otherwise

Question: Which columns would you select and avoid selecting as features, and why? Keep in mind the goal is to identify features that will serve as useful predictors for the target variable, `target_5yrs`.

- You should avoid selecting the `name` column as a feature. A player's name is not helpful in determining their career duration. Moreover, it may not be ethical or fair to predict a player's career duration based on a name.
- The number of games a player has played in may not be as important in determining their career duration as the number of points they have earned. While you could say that someone who has played in more games may have more practice and experience, the points they earn during the games they played in would speak more to their performance as a player. This, in turn, would influence their career duration. So, the `gp` column on its own may not be a helpful feature. However, `gp` and `pts` could be combined to get the total number of points earned across the games played, and that result could be a helpful feature. That approach

can be implemented later in the feature engineering process—in feature extraction.

- If the number of points earned across games will be extracted as a feature, then that could be combined with the number of minutes played across games (`min * gp`) to extract another feature. This could be a measure of players' efficiency and could help in predicting players' career duration. `min` on its own may not be useful as a feature for the same reason as `gp`.
- There are three different columns that give information about field goals. The percent of field goals a player makes (`fg`) says more about their performance than the number of field goals they make (`fgm`) or the number of field goals they attempt (`fga`). The percent gives more context, as it takes into account both how many field goals a player successfully made and how many field goals they attempted in total. This allows for a more meaningful comparison between players. The same logic applies to the percent of three-point field goals made, as well as the percent of free throws made.
- There are columns for the number offensive rebounds (`oreb`), the number of defensive rebounds (`dreb`), and the number of rebounds overall (`reb`). Because the overall number of rebounds should already incorporate both offensive and defensive rebounds, it would make sense to use the overall as a feature.
- The number of assists (`ast`), steals (`stl`), blocks (`blk`), and turnovers (`tov`) also provide information about how well players are performing in games, and thus, could be helpful in predicting how long players last in the league.

Therefore, at this stage of the feature engineering process, it would be most effective to select the following columns:

`gp`, `min`, `pts`, `fg`, `3p`, `ft`, `reb`, `ast`, `stl`, `blk`, `tov`.

Next, select the columns you want to proceed with. Make sure to include the target column, `target_5yrs`. Display the first few rows to confirm they are as expected.

[9]:
```
# Select the columns to proceed with and save the DataFrame in new variable␣
 ↪`selected_data`.
# Include the target column, `target_5yrs`.

### YOUR CODE HERE ###

selected_data = data[["gp", "min", "pts", "fg", "3p", "ft", "reb", "ast",␣
 ↪"stl", "blk", "tov", "target_5yrs"]]


# Display the first few rows.

### YOUR CODE HERE ###


selected_data.head()
```

[9]:
```
   gp   min  pts    fg    3p    ft  reb  ast  stl  blk  tov  target_5yrs
0  36  27.4  7.4  34.7  25.0  69.9  4.1  1.9  0.4  0.4  1.3            0
1  35  26.9  7.2  29.6  23.5  76.5  2.4  3.7  1.1  0.5  1.6            0
2  74  15.3  5.2  42.2  24.4  67.0  2.2  1.0  0.5  0.3  1.0            0
```

```
3  58  11.6  5.7  42.6  22.6  68.9  1.9  0.8  0.6  0.1  1.0            1
4  48  11.5  4.5  52.4   0.0  67.4  2.5  0.3  0.3  0.4  0.8            1
```

Hint 1

Refer to the materials about feature selection and selecting a subset of a DataFrame.

Hint 2

Use two pairs of square brackets, and place the names of the columns you want to select inside the innermost brackets.

Hint 3

There is a function in `pandas` that can be used to display the first few rows of a DataFrame. Make sure to specify the column names with spelling that matches what's in the data. Use quotes to represent each column name as a string.

1.5.2 Feature transformation

An important aspect of feature transformation is feature encoding. If there are categorical columns that you would want to use as features, those columns should be transformed to be numerical. This technique is also known as feature encoding.

Question: Why is feature transformation important to consider? Are there any transformations necessary for the features you want to use? - Many types of models are designed in a way that requires the data coming in to be numerical. So, transforming categorical features into numerical features is an important step. - In this particular dataset, `name` is the only categorical column and the other columns are numerical (discussed in the exemplar response to Question 2). Given that `name` is not selected as a feature, all of the features that are selected at this point are already numerical and do not require transformation.

1.5.3 Feature extraction

Display the first few rows containing containing descriptions of the data for reference. The table is as follows:

Column Name	Column Description
`name`	Name of NBA player
`gp`	Number of games played
`min`	Number of minutes played per game
`pts`	Average number of points per game
`fgm`	Average number of field goals made per game
`fga`	Average number of field goal attempts per game
`fg`	Average percent of field goals made per game
`3p_made`	Average number of three-point field goals made per game
`3pa`	Average number of three-point field goal attempts per game
`3p`	Average percent of three-point field goals made per game
`ftm`	Average number of free throws made per game

Column Name	Column Description
`fta`	Average number of free throw attempts per game
`ft`	Average percent of free throws made per game
`oreb`	Average number of offensive rebounds per game
`dreb`	Average number of defensive rebounds per game
`reb`	Average number of rebounds per game
`ast`	Average number of assists per game
`stl`	Average number of steals per game
`blk`	Average number of blocks per game
`tov`	Average number of turnovers per game
`target_5yrs`	1 if career duration >= 5 yrs, 0 otherwise

[10]:
```
# Display the first few rows of `selected_data` for reference.

### YOUR CODE HERE ###

selected_data.head()
```

[10]:
```
   gp   min  pts    fg    3p    ft  reb  ast  stl  blk  tov  target_5yrs
0  36  27.4  7.4  34.7  25.0  69.9  4.1  1.9  0.4  0.4  1.3            0
1  35  26.9  7.2  29.6  23.5  76.5  2.4  3.7  1.1  0.5  1.6            0
2  74  15.3  5.2  42.2  24.4  67.0  2.2  1.0  0.5  0.3  1.0            0
3  58  11.6  5.7  42.6  22.6  68.9  1.9  0.8  0.6  0.1  1.0            1
4  48  11.5  4.5  52.4   0.0  67.4  2.5  0.3  0.3  0.4  0.8            1
```

Question: Which columns lend themselves to feature extraction?

- The `gp`, `pts`, `min` columns lend themselves to feature extraction.
  - `gp` represents the total number of games a player has played in, and `pts` represents the average number of points the player has earned per game. It might be helpful to combine these columns to get the total number of points the player has earned across the games and use the result as a new feature, which could be added into a new column named `total_points`. The total points earned by a player can reflect their performance and shape their career longevity.
  - The `min` column represents the average number of minutes played per game. `total_points` could be combined with `min` and `gp` to extract a new feature: points earned per minute. This can be considered a measure of player efficiency, which could shape career duration. This feature can be added into a column named `efficiency`.

Extract two features that you think would help predict `target_5yrs`. Then, create a new variable named 'extracted_data' that contains features from 'selected_data', as well as the features being extracted.

[11]:
```
# Extract two features that would help predict target_5yrs.
# Create a new variable named `extracted_data`.

### YOUR CODE HERE ###
```

```
# Make a copy of `selected_data`
extracted_data = selected_data.copy()

# Add a new column named `total_points`;
# Calculate total points earned by multiplying the number of games played by␣
↪the average number of points earned per game
extracted_data["total_points"] = extracted_data["gp"] * extracted_data["pts"]

# Add a new column named `efficiency`. Calculate efficiency by dividing the␣
↪total points earned by the total number
# of minutes played, which yields points per minute. (Note that `min`␣
↪represents avg. minutes per game.)
extracted_data["efficiency"] = extracted_data["total_points"] /␣
↪(extracted_data["min"] * extracted_data["gp"])

# Display the first few rows of `extracted_data` to confirm that the new␣
↪columns were added.
extracted_data.head()
```

[11]:

```
   gp   min  pts    fg    3p    ft  reb  ast  stl  blk  tov  target_5yrs  \
0  36  27.4  7.4  34.7  25.0  69.9  4.1  1.9  0.4  0.4  1.3            0
1  35  26.9  7.2  29.6  23.5  76.5  2.4  3.7  1.1  0.5  1.6            0
2  74  15.3  5.2  42.2  24.4  67.0  2.2  1.0  0.5  0.3  1.0            0
3  58  11.6  5.7  42.6  22.6  68.9  1.9  0.8  0.6  0.1  1.0            1
4  48  11.5  4.5  52.4   0.0  67.4  2.5  0.3  0.3  0.4  0.8            1

   total_points  efficiency
0         266.4    0.270073
1         252.0    0.267658
2         384.8    0.339869
3         330.6    0.491379
4         216.0    0.391304
```

Hint 1

Refer to the materials about feature extraction.

Hint 2

Use the function `copy()` to make a copy of a DataFrame. To access a specific column from a DataFrame, use a pair of square brackets and place the name of the column as a string inside the brackets.

Hint 3

Use a pair of square brackets to create a new column in a DataFrame. The columns in DataFrames are series objects, which support elementwise operations such as multiplication and division. Be sure the column names referenced in your code match the spelling of what's in the DataFrame.

Now, to prepare for the Naive Bayes model that you will build in a later lab, clean the extracted data and ensure ensure it is concise. Naive Bayes involves an assumption that features are independent of each other given the class. In order to satisfy that criteria, if certain features are aggregated to yield new features, it may be necessary to remove those original features. Therefore, drop the columns that were used to extract new features.

Note: There are other types of models that do not involve independence assumptions, so this would not be required in those instances. In fact, keeping the original features may be beneficial.

[12]:
```
# Remove any columns from `extracted_data` that are no longer needed.

### YOUR CODE HERE ###

# Remove `gp`, `pts`, and `min` from `extracted_data`.
extracted_data = extracted_data.drop(columns=["gp", "pts", "min"])

# Display the first few rows of `extracted_data` to ensure that column drops␣
↪took place.

### YOUR CODE HERE ###

extracted_data.head()
```

[12]:
```
     fg    3p    ft  reb  ast  stl  blk  tov  target_5yrs  total_points  \
0  34.7  25.0  69.9  4.1  1.9  0.4  0.4  1.3            0         266.4
1  29.6  23.5  76.5  2.4  3.7  1.1  0.5  1.6            0         252.0
2  42.2  24.4  67.0  2.2  1.0  0.5  0.3  1.0            0         384.8
3  42.6  22.6  68.9  1.9  0.8  0.6  0.1  1.0            1         330.6
4  52.4   0.0  67.4  2.5  0.3  0.3  0.4  0.8            1         216.0

   efficiency
0    0.270073
1    0.267658
2    0.339869
3    0.491379
4    0.391304
```

Hint 1

Refer to the materials about feature extraction.

Hint 2

There are functions in the `pandas` library that remove specific columns from a DataFrame and that display the first few rows of a DataFrame.

Hint 3

Use the `drop()` function and pass in a list of the names of the columns you want to remove. By default, calling this function will result in a new DataFrame that reflects the changes you made.

The original DataFrame is not automatically altered. You can reassign `extracted_data` to the result, in order to update it.

Use the `head()` function to display the first few rows of a DataFrame.

Next, export the extracted data as a new .csv file. You will use this in a later lab.

```
[13]: # Export the extracted data.

### YOUR CODE HERE ###

extracted_data.to_csv("extracted_nba_players_data.csv", index=0)
```

Hint 1

There is a function in the `pandas` library that exports a DataFrame as a .csv file.

Hint 2

Use the `to_csv()` function to export the DataFrame as a .csv file.

Hint 3

Call the `to_csv()` function on `extracted_data`, and pass in the name that you want to give to the resulting .csv file. Specify the file name as a string and in the file name. Make sure to include `.csv` as the file extension.

```
Also, pass in the parameter `index` set to `0`, so that when the export occurs, the row indice
```

1.6 Considerations

What are some key takeaways that you learned during this lab? - It is important to check for class balance in a dataset, particularly in the context of feature engineering and predictive modeling. If the target column in a dataset has more than 90% of its values belonging to one class, it is recommended to redistribute the data; otherwise, once a model is trained on the imbalanced data and predictions are made, the predictions may be biased. - Feature selection involves choosing features that help predict the target variable and removing columns that may not be helpful for prediction. In this process, and throughout feature engineering, it is important to make ethical considerations.
- Feature transformation involves transforming features so that they are more usable for future modeling purposes, which includes encoding categorical features to turn them into numerical features. - Feature extraction involves combining existing columns meaningfully to construct new features that would help improve prediction.

What summary would you provide to stakeholders? Consider key attributes to be shared from the data, as well as upcoming project plans. - The following attributes about player performance could help predict their NBA career duration and should be included in a presentation to stakeholders: field goals, three-point field goals, free throws, rebounds, assists, steals, blocks, turnovers, total points, and efficiency as points per minute. - It would be important to explain that these attributes, along with a relevant dataset, will be used in the next stage of the project. At that point, a model will be built to predict a player's career duration. Insights gained will be shared with stakeholders once the project is complete. Stakeholders would also appreciate being provided with a timeline and key deliverables that they can expect to receive.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged