Assignment 03 - AI Prompts

Gavin Boss

September 20, 2025

# 1. Appendix 1

|  |
| --- |
| Tip |
| Prompt  ————————————————————————— ValueError Traceback (most recent call last) Cell In[34], line 25 22 plt.ylabel(“Feature Name”) 23 plt.tight\_layout() —> 25 featureimportanceplot\_df= pd.DataFrame({ 26 “Feature”: feature\_names, 27 “Importance”: importances 28 }).sort\_values(by=“Importance”, ascending=False) 30 # Save and show 31 plt.savefig(“\_output/rf\_feature\_importance.png”, dpi=300) File ~/assignment-04-gavinboss/.venv/lib/python3.12/site-packages/pandas/core/frame.py:782, in DataFrame.**init**(self, data, index, columns, dtype, copy) 776 mgr = self.\_init\_mgr( 777 data, axes={“index”: index, “columns”: columns}, dtype=dtype, copy=copy 778 ) 780 elif isinstance(data, dict): 781 # GH#38939 de facto copy defaults to False only in non-dict cases –> 782 mgr = dict\_to\_mgr(data, index, columns, dtype=dtype, copy=copy, typ=manager) 783 elif isinstance(data, ma.MaskedArray): 784 from numpy.ma import mrecords File ~/assignment-04-gavinboss/.venv/lib/python3.12/site-packages/pandas/core/internals/construction.py:503, in dict\_to\_mgr(data, index, columns, dtype, typ, copy) 499 else: 500 # dtype check to exclude e.g. range objects, scalars … 680 raise ValueError( 681 “Mixing dicts with non-Series may lead to ambiguous ordering.” 682 ) ValueError: All arrays must be of the same length |

|  |
| --- |
| Tip |
| Ah, I see exactly what’s happening here. The error:  ValueError: All arrays must be of the same length  occurs because when you create a DataFrame with a dictionary like this:  featureimportanceplot\_df = pd.DataFrame({  "Feature": feature\_names,  "Importance": importances }).sort\_values(by="Importance", ascending=False)  **feature\_names and importances do not have the same length**. Pandas requires that each column array be exactly the same length. 1.0.1 How to fix it  1. **Check the lengths**:   print(len(feature\_names)) print(len(importances))  If they don’t match, that’s the root cause.   1. **Make them match**:  * If you are using a RandomForestRegressor or RandomForestClassifier, you typically get the importances via:   importances = model.feature\_importances\_   * The number of features (feature\_names) must exactly match the length of importances. For example, if you used a ColumnTransformer or OneHotEncoder, the transformed feature array may have more columns than your original feature\_names. * To fix, either:   + Update feature\_names to reflect all columns after preprocessing (including one-hot encoding).   + Or use only the features that match the importances array.   A quick sanity check:  print("Feature Names:", len(feature\_names)) print("Importances:", len(importances))  If feature\_names is shorter, it’s probably because you didn’t expand categorical variables after one-hot encoding.  You can generate the correct names like this if you used OneHotEncoder:  from sklearn.compose import ColumnTransformer  # Assuming ct is your ColumnTransformer feature\_names = ct.get\_feature\_names\_out()  If you want, I can rewrite your featureimportanceplot\_df snippet so it **works even after one-hot encoding**, and also generates a clean plot.  Do you want me to do that? |