Making Predictions with Trained Models

Wahoo!

What will be covered today

- Shout out to Blake- his question spurred this on!
- How to make predictions with your model once you trained it

Housekeeping

- MP3 Due Date
- Clarification about what data cleaning you have to do
  - Have to check for missing values and outliers
    - If there aren’t any that’s ok!
  - Must do encoding and features scaling

Making Predictions from a trained model

- Typically, you save your model onto your computer!
- Then you can load it anytime and use it to make predictions

Pickle Library

- Used to save and load data structures within a Python environment
- Can be used to save/load models and transformers

Saving Files with Pickle Library

```
with open('cityHappinessNN.pkl', 'wb') as f:
    pickle.dump(nnModel, f)
```

- Talk about with open files
  - How do they work?
  - What do variables mean

Pickle Library

- r- means read
- w- means write
- rb- means read a binary file
- wb- means write a binary file

Loading Files with the Pickle Library

```
with open('cityHappinessNN.pkl', 'rb') as f:
    loadedNNModel = pickle.load(f)
```

Open Starter Code

- Should have notebook that made some models for predicting city happiness
- Review what the code does
- New notebook to make predictions
- This split is nice.

Saving your models using pickle

- Do this together, lets save all 3 ml models
  - Not linear regression- it was trash

Are we ready to load in our models and make some predictions?

- Is our prediction dataframe ready?

We’re not Ready!

- Our prediction needs to be converted to the same format that our models expect!
  - What does this entail?

Data Cleaning Pipeline

- Handling Missing Values
- Encoding Categorical Data
- Removing Outliers
- Feature Scaling
- What should we do to our prediction data?

We should

- Encode our features
  - Label encoding and
  - One Hot Encoding
- Feature Scaling
  - Normalization or Standardization

Lets begin!

- Label Encoding
  - We made a custom protocol for this
  - Let’s do the same here

One Hot Encoding

- We used sklearn’s library for this.
- How can we make sure that our prediction dataframe is transformed exactly like the original?
  - What do you think?

One Hot Encoding

- We can save and reload the original encoder using pickle!
- Lets do this!

Common Methods in sklearn

- fit()
- transform()
- fit_transform()
- predict()
- What do you think each of these does?

Common Methods in sklearn

- fit()
  - Changes an object based on the given data
- transform()
  - Applies a transformation based on how it was fit
- fit_transform()
  - Does a fit and a transform at the same time
- predict()
  - Makes a prediction

Ex: One Hot encoding

- In our training file, we use fit_transform to fit the encoder and then also apply it to our features
- In our prediction file, we only use transform to apply
- Do example on whiteboard
  - If we fit our encoder on our one prediction point, what would it do? Is this what we want?

Next Step

- Lets save, load, and apply our encoder to our prediction point. Let’s confirm everything looks ok.

Next step- feature scaling

- What do you think we will do with this?

Next step- feature scaling

- What do you think we will do with this?
  - We will also save, load, and apply the transform method for this too.

ERROR

- When applying the transform, it expects to see one more column then we are giving it.
  - Why? What is this extra column?

It is wanting the Happiness Score!

- Why?
  - Because when we fit it, it was given the happiness score.
  - How do we fix this?

It is wanting the Happiness Score!

- How do we fix this?
  - We have to separate out the target variable before we standardize
  - In other words, after we remove outliers, we need to separate the features and target, and the scale the features and target separately

More benefits of splitting features and target before scaling

- Data Leakage
  - Does anyone know what this means?

More benefits of splitting features and target before scaling

- Data Leakage
  - This is when information from the testing set somehow “leaks” into the training of your model.
  - Allows the model to “cheat” so that it may perform better on the testing set then it otherwise would.

More benefits of splitting features and target before scaling

- If we feature scale with our full dataset (training and testing sets)- they will include the testing data in the scaling.
- This is not a huge deal, but it is still a bit helpful to split the data into a training and testing set, and then scale the features.
    - No drawbacks to doing this

Let’s do this

- Change training notebook to do the scaling after the data is split
- We will have to scale the x and y data separately with their own scalers.
- On the training data, do we fit, transform, or fit_transform?
- On the testing data, do we fit, transform, or fit_transform?

Applying Scalers to Prediction

- Using Pickle, save and load the feature scaler and apply it to the prediction datapoint
- Confirm that it looks scaled

Prediction Time!

- Now we can make predictions!!
- Load the different models
  - Use the predict method.
- HOWEVER, there is one more issue…

Models were trained on Scaled Target

- That means, the prediction of our model will be a scaled answer
  - If standardscaler was used, it would have an avg of 0 and a std of 1.
- How do we get back to the original scale?

inverse_transform!

- Load the saved yScaler (targetScaler)
- Use the inverse_transform method
  - And we’re good!

We should be able to make predictions now

- Test this
- What happens when we change our features?
- Notice anything weird?
  - Anything weird about decision tree? Why?