I followed along with the Udemy course using the Spyer IDE for datascience. It's missing some basic features like code completion but there's one feature it does have that I really like. A variable explorer tab that will give you a visual representation of the variables defined in your workspace.

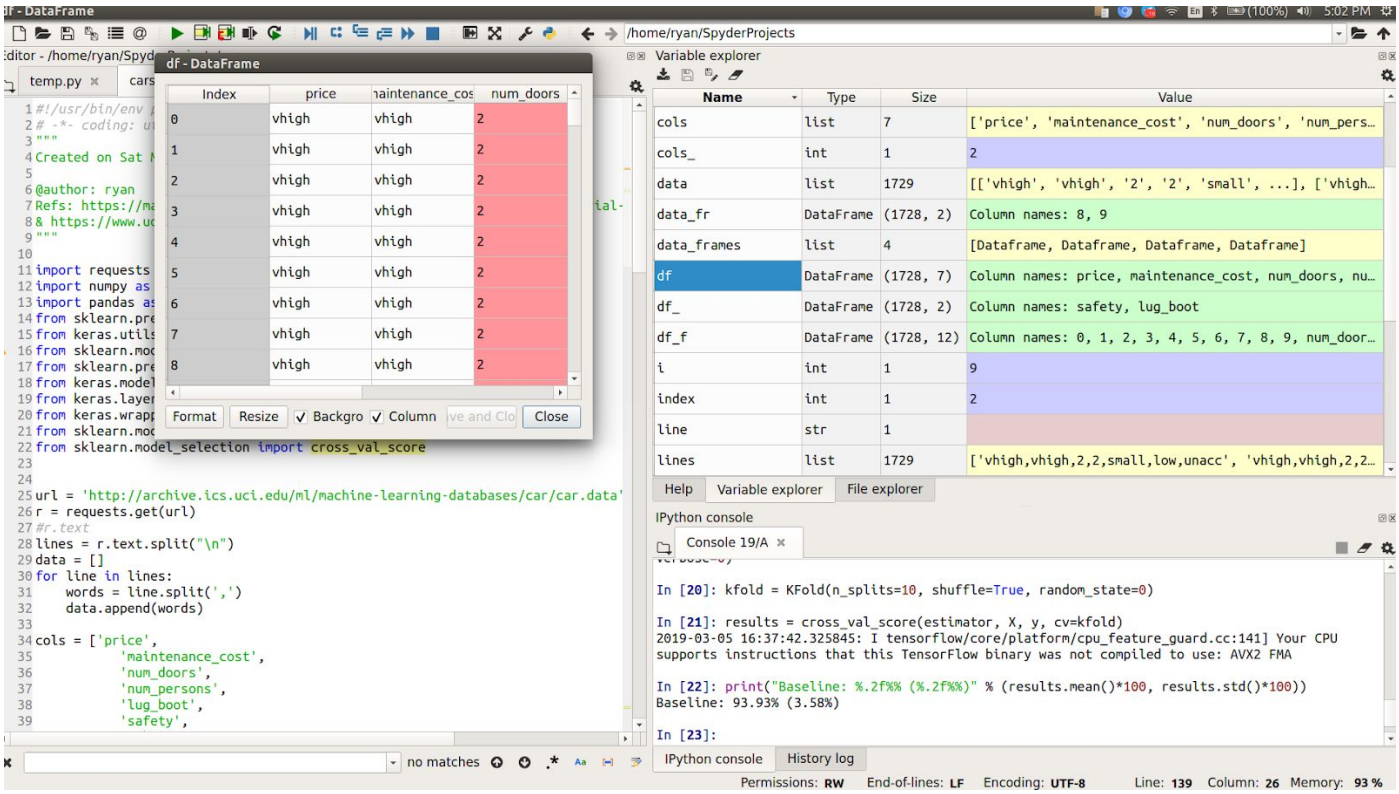

Figure 1

It might be a bit hard to see but basically you have your editor on the left, an Ipython console on the right as well as the Variable Explorer I mentioned. I had just clicked on the variable df and it shows a visual representation of that dataframe. Very very handy. Also with functions there's a way you can get documentation for them with info on parameters for each function which came in very handy as well.

I followed the code from the example, see appendix, pretty closely but needed to do considerably more data pre-processing. The reason being I had multiple categorical features that needed to be one hot encoded. In the example provided there was just one feature that needed this and it was easy to figure out how to drop the extra dummy variable to avoid the dummy variable trap.

In my case it was more difficult. First I tried to one hot encode each column one at a time but the one_hot_encoder() threw errors when some of the columns were still categorical.

I decided to break each column of the df off into its own dataframe to do the one hot encoded but again I ran into a problem either the LabelEncoder or OneHotEncoder which threw an error since it was expecting a 2D structure but was getting a list. My solution was to grab two columns from the original df, the column of interest and another extra column. I made deep copies to not affect the original df. Then after one hot encoding that I would drop a dummy column and that other extra column, then add this to a list to be concatenated later to form a complete df with one hot encoding but avoid the dummy variable trap.

I ran into further problems mostly having to do with the fact that the example code I was following had a label that was a boolean where my label was categorical so I found another example online that more closely matched my situation.

https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

The code I have written is combination of the Udemy course and that tutorial with some custom data pre-processing by yours truly.

I was able to achieve an accuracy of 93.3% with a std of 3.58% usig k-folds validation and cross_val_score().

All in all I would consider this a success for a first try. I will be looking further into other deep learning models and their application with regards to Sentiment Analysis.

I have a strong interest in Sentiment Analysis since I do some investing in my spare time and see great potential there.

Refs:

https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/
https://www.udemy.com/deeplearning/

Appendix:

Example code I was working off of from Udemy:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])

labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_1.fit_transform(X[:, 2])

onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Part 2 Making the ANN!
```

```
# import the keras library
from keras.models import Sequential
from keras.layers import Dense

# Initialize the ANN
# define as a sequence of layers
classifier = Sequential()

# input layer and first hidden layer
classifier.add(Dense(input_dim=11, activation="relu", units=6, kernel_initializer="uniform"))


# adding the second hidden layer
classifier.add(Dense(activation="relu", units=6, kernel_initializer="uniform"))

# adding the output layer
classifier.add(Dense(activation="sigmoid", units=1, kernel_initializer="uniform"))

# compliling the ANN
# adam is a type of stochastic gradient decent
classifier.compile(optimizer='adam', loss= 'binary_crossentropy', metrics= ['accuracy'])

# Fitting the ANN to the training set
classifier.fit(X_train, y_train, batch_size=10, epochs=100)

# Fitting classifier to the Training set
# Create your classifier here

# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)



# Homework 1
"""Predict if the customer with this info will leave the bank:
Geography: France
Credit Score: 600
Gender: Male
Age: 40
Tenure: 3
Balance: 60000
Number Products: 2
```

```
Has Credit Card: yes
Is Active Member: yes
Estimated Salary: 50000
"""
new_prediction = classifier.predict(sc.transform(np.array([[0.0, 0, 600, 1, 40, 3, 60000, 2, 1, 1,
50000 ]])))
new_prediction = (new_prediction > .5)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```