

# Applied Machine Learning, II

---

INFO 370

# Learning Objectives

Review predicting blood donations

Identify data challenges for assignment 4:

- Data is large
- Data is missing
- Some features are categorical
- Different predictions for each country
- Individual data needs to be incorporated

# Predicting Blood Donations

---

What was hard //  
time consuming?

How can you  
minimize steps that  
are hard / time  
consuming?

# Don't Repeat Yourself

For every model you want to run, you'll need to

- **Load** the data
- **Clean** the data
- Import / create a **classifier**
- Create a **pipeline** that does a **grid search** for the best parameters through **cross validation** (including pre-processing)
- Generate, format, and save a set of **predictions** for your test data
- In a4, you'll need to do this for **each country**

**The only things that change are you model and your parameters**

# Exploring Models

In this Notebook, I experiment with some models before doing a parameter search to get a sense of the performance of each model.

```
In [4]: # Import models and helper functions
import helper_functions as hf
import importlib
importlib.reload(hf)
import numpy as np
```

## Random Forest

As a first pass at the data, try a `RandomForestClassifier` *without* optimizing any parameters (for now).

```
In [7]: # Import model, then pass the model to our function -- this will do **all steps** for **all countries**
from sklearn.ensemble import RandomForestClassifier
hf.model_all_countries(model=RandomForestClassifier(), grid={}, name="test-rf")
```

```
Country A
Log Loss: 0.539508912379
Country B
Log Loss: 0.899435881764
Country C
Log Loss: 0.196875362179
Average Log Loss for test-rf: 0.545273385441
```

## Decision Tree

What a4 could (should) look like

# Data Challenges

---



# The Data is Large

It will take you a long time to load/prep the data

Your models may take a long time to run

Tuning your parameters may take a long time (too long!)

Be patient and prepared for this

# Dimensionality Reduction

One approach is to reduce the number of features

Combine multiple features to create new ones

Multiple approaches for this, including Principal Component Analysis

Easily built into your pipeline:

```
if(do_pca == True):  
    pca = PCA(n_components=10)  
    pipe = make_pipeline(scaler, pca, clf)
```

Not necessarily required, but might be necessary for some models (based on time)

```

pca.fit(X)
X = pca.transform(X)

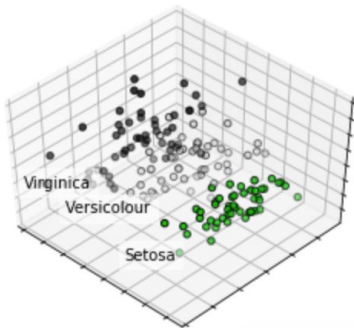
for name, label in [('Setosa', 0), ('Versicolour', 1), ('Virginica', 2)]:
    ax.text3D(X[y == label, 0].mean(),
              X[y == label, 1].mean() + 1.5,
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(np.float)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.spectral,
           edgecolor='k')

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])

plt.show()

```

Automatically created module for IPython interactive environment



# Some Data is Missing

What is a reasonable conceptual approach?

- Drop missing rows
- Replace missing values as 0
- Replace missing values as the mean / median / mode

Easily incorporated into your pipeline

```
from sklearn.preprocessing import Imputer  
imputer=Imputer()  
pipe = make_pipeline(imputer, scaler, clf)
```

# Some Features are Categorical


So what?

We already have to work around this

	<b>coef</b>	<b>std err</b>	<b>t</b>	<b>P&gt; t </b>	<b>[0.025</b>	<b>0.975]</b>
<b>Intercept</b>	1.637e+04	757.573	21.615	0.000	1.49e+04	1.79e+04
<b>sx[T.male]</b>	-505.6541	843.489	-0.599	0.552	-2203.508	1192.200
<b>rk[T.associate]</b>	4253.7176	961.967	4.422	0.000	2317.379	6190.056
<b>rk[T.full]</b>	9508.1038	923.022	10.301	0.000	7650.157	1.14e+04
<b>dg[T.masters]</b>	314.4186	782.166	0.402	0.690	-1260.000	1888.837
<b>yr</b>	385.1491	77.415	4.975	0.000	229.322	540.977

	sx	rk	yr	dg	yd	sl
0	male	full	25	doctorate	35	36350
1	male	full	13	doctorate	22	35350
2	male	full	10	doctorate	23	28200
3	female	full	7	doctorate	27	26775
4	male	full	19	masters	30	33696
5	male	full	16	doctorate	21	28516
6	female	full	0	masters	32	24900
7	male	full	16	doctorate	18	31909
8	male	full	13	masters	30	31850
9	male	full	13	masters	31	32850
10	male	full	12	doctorate	22	27025

```
dummy_data = pd.get_dummies(salary_data)
```



	yr	yd	sl	sx_female	sx_male	rk_assistant	rk_associate	rk_full	dg_doctorate	dg_masters
0	25	35	36350	0	1	0	0	1	1	0
1	13	22	35350	0	1	0	0	1	1	0
2	10	23	28200	0	1	0	0	1	1	0
3	7	27	26775	1	0	0	0	1	1	0
4	19	30	33696	0	1	0	0	1	0	1
5	16	21	28516	0	1	0	0	1	1	0
6	0	32	24900	1	0	0	0	1	0	1
7	16	18	31909	0	1	0	0	1	1	0
8	13	30	31850	0	1	0	0	1	0	1
9	13	31	32850	0	1	0	0	1	0	1
10	12	22	27025	0	1	0	0	1	1	0
11	15	19	24750	0	1	0	1	0	1	0
12	9	17	28200	0	1	0	0	1	1	0
13	9	27	23712	0	1	0	1	0	0	1

Transform categorical data into dummy variables

# Different Data Sets for Each Country

You'll want to either incorporate country as a predictor, or generate different models by country

Managing these files can be cumbersome

The [benchmark code](#) provides a great reference

The best strategy  
will be to have a well  
organized project.



# Individual Data

Predictions are made at the **household level**

You are **required** to incorporate individual data

How can/should you incorporate features from individuals for each household?

# Summary

Make sure you:

- Write **helper functions** to have a clean and **organized project**!
- Incorporate **individual data**
- Generate **dummy features** from categorical features
- **Impute** missing values
- Use a **Scaler** on your data
- Use **cross-validation** to search for optimal model parameters for you classifier
- Hint: use the **scorer** to match the competition scorer

The best strategy  
will be to have a well  
organized project.

# Upcoming...

Assignment-4 due **next Wednesday**

Start chipping away at **your final projects (due in 2 weeks!)**