Identifying safe loans with decision trees

The LendingClub is a peer-to-peer leading company that directly connects borrowers and potential lenders/investors. In this notebook, you will build a classification model to predict whether or not a loan provided by LendingClub is likely to default.

In this notebook you will use data from the LendingClub to predict whether a loan will be paid off in full or the loan will be charged off and possibly go into default. In this assignment you will:

- Use SFrames to do some feature engineering.
- Train a decision-tree on the LendingClub dataset.
- Visualize the tree.
- Predict whether a loan will default along with prediction probabilities (on a validation set).
- Train a complex tree model and compare it to simple tree model.

If you are doing the assignment with IPython Notebook

An IPython Notebook has been provided below to you for this assignment. This notebook contains the instructions, quiz questions and partially-completed code for you to use as well as some cells to test your code.

What you need to download

If you are using GraphLab Create:

- Download the Lending club data in SFrame format: lending-club-data.gl.zip (https://eventing.coursera.org/api/redirectStrict/1mTzQ-3WiUZP_y6_3JA2mHg_5XnHBU5WFqfPiVvHwcclXTNJ_CW_j90tCgB43gR-EPCzs_eaXRCYCnAyNnx9LQ.KYveSFGVkqZpfGZtnd6nLA.zihlV-ElC2U-EQAsYRbQldrNQ6VK68bHZZ7bu79S9ROMVz5p-9WnGIXKa8iAbdUKPc95eYNJO9RcSoqpKMhN9jbmaFlIkXThFu1c9Y0XbQNbQw6gzkatRhWc4bSEF3BAsdG3KBpEekbU7lWrQ8F18DXWARp6wm-C7rFX7lKuCUmJcluwnMu85XCOCJFmnMCNEKwZ1cbULKoWagZTKkoEf8nK0Qc6XCEMGBU1bVf1Ged-CrIaPPE-Z-gskrtw8XNQ0_hGJMDy-ETVmaEubGS6YwcVvAoosVycD5yM_LQSt6rlqxpapG-TASUq5O6Qa-KCt60GYpf5liHVq5acqt5xKvXHZDm74mzZkm2EBYLXvh0jJjmZdqECCXg99n8nzlp2dUd3NnlpaUNP0l_sau9KW2u6O3AkTiSmu_Ygia1G2ksXW6SMK-5KS7H0UhTWPoFM)
- Download the companion IPython Notebook: module-5-decision-tree-assignment-1-blank.ipynb (https://eventing.coursera.org/api/redirectStrict/GlmH1k3WNlSRogShe4VhwtwIUtgQ1iRMxgcgcM3p7tdsfYaO-0MUoeErpjIQL27WGsPWiXIS0Lndd3IzX5A38A.7i5yHz0DfGKMuVXSXgBqmQ.yGAyJQMREeK23_2ys3hoxE1jAb2rwAZ_ZGSgDdPH6pVUw3B2PZX5HYQT_923jgrODIiMNWs9O932BZvn9dEfaVP2-Y4Vl-yd4hIEgw-cCn4ynt_cf8-HYPadqCYOcnZMpsw6JypeZhsMHBcihQl5oUZdqHascMDmlcLIfvCBmUue4GKObTsJ5VttwdszKMAlOem77iyf1jKDcW4PSbUQDPFUc2atPrgOMeCAzdXlBJZq5gJJa33Z-DgKcAukJcYI-CDhVBNfm6Oxn2D40Qrk_EUuAfc806KuKpvro7LqR1A9L-EPqx-pD2WcO5JMLyxXGfeT3WElygOJc0uhwhn2lYJ0WTdJPEExgK7nND9Wx9kBMKjP9wx5ScL8eEFcPPLk4UlaGOzaqq-tPDeDycwPu_YP8mMJ6YNUcC8xpzj45M7YFcLhTxfHLnuFUVJMYoc5cdouz7h4nfsL-ItU9cxlU1VhOIQVL6A7Qc-Q0tbm2TmKH3DUMiDQHvkYant9paGW)
- Save both of these files in the same directory (where you are calling IPython notebook from) and unzip the data file.

- Follow the instructions contained in the IPython notebook.

If you are not using GraphLab Create:

- If you are using SFrame, download the LendingClub dataset in SFrame format: lending-club-data.gl.zip (https://eventing.coursera.org/api/redirectStrict/1mTzQ-3WiUZP_y6_3JA2mHg_5XnHBU5WFqfPiVvHwcclXTNJ_CW_j90tCgB43gR-EPCzs_eaXRCYCnAyNnx9LQ.KYveSFGVkqZpfGZtnd6nLA.zihlV-ElC2U-EQAsYRbQldrNQ6VK68bHZZ7bu79S9ROMVz5p-9WnGIXKa8iAbdUKPc95eYNJO9RcSoqpKMhN9jbmaFlIkXThFu1c9Y0XbQNbQw6gzkatRhWc4bSEF3BAsdG3KBpEekbU7lWrQ8F18DXWARp6wm-C7rFX7lKuCUmJcIuwnMu85XCOCJFmnMCNEKwZ1cbULKoWagZTKkoEf8nK0Qc6XCEMGBU1bVf1Ged-CrIaPPE-Z-gskrtw8XNQ0_hGJMDy-ETVmaEubGS6YwcVvAoosVycD5yM_LQSt6rlqxpapG-TASUq5O6Qa-KCt60GYpf5liHVq5acqt5xKvXHZDm74mzZkm2EBYLXvh0jJjmZdqECCXg99n8nzlp2dUd3NnlpaUNP0l_sau9KW2u6O3AkTiSmu_Ygia1G2ksXW6SMK-5KS7H0UhTWPoFM)
- If you are using a different package, download the LendingClub dataset in CSV format: lending-club-data.csv.zip (https://eventing.coursera.org/api/redirectStrict/04Rcq6u0NCoNKp-TwFPjfA36mlKE19r7KP-M-PHZC731_1wlgiVo9metEcSECZ8YYO0YKTr5d106sgj8VW0Efg.OFDEAupXG5sWDYE8_b7puQ.95u2wNM2TNi7kLx5_LdIGD75UFNuMrTrqyROh4uCx-vD1-d9QWOAfa6YN7n26unIaRutr3ou_J4n1_67N9CsHclt5BjSVDghIQ_1aV6ZXfeQPQU-WF-DrcG6EXzl4-EwCs67KajVU2YCmhEv-pw3ShmZsYVVLb7pgFc2RpxdeuWC9ap3y79jsiebXZ5dqcHZFBJC2BNDubUDa8tx-1tdUXNsob6wp_d_0FOA-FsNVHAsiF2M8P-X0eIxBkP9Cuw5f4jNKgG0-bai1LRNKJh8lPgyW3ujCOXL_5AZXCzWYtxqdd4O7k47ska9xMdIOYdShIEAx--YDksnojRpQYv54-82Mu5oP7E_r0oP2sfT8rVxvi8UckVDvwTMl-ats8C_R6B738jnJDr2GQqS83a9auWqndhkrNQqGyR4JacaCGb3n1qbiVWtSpND2zGahtIs)

If you are using GraphLab Create and the companion IPython Notebook

Open the companion IPython notebook and follow the instructions in the notebook.

If you are using other tools

This section is designed for people using tools other than GraphLab Create. Even though some instructions are specific to scikit-learn, most part of the assignment should be applicable to other tools as well. However, we highly suggest you use SFrame (https://eventing.coursera.org/api/redirectStrict/ph6BCG0IKG3nIlpUZDEeKS4n_2R0iqtK-uV7P5TwcglBc8OgSpVE-TEoZzxzu2K_boDyCuDGzEF0EvRTP3E1XQ.LiSzP8j9ounO-uDmasImZQ.UpVBG9X5mftzkGkePu4jcrmoy6X25MraPlnL5gfYr3xSuDJ6x3FM7KulUCTroWgqQHe5N_bj_5k0qgv2v_bsCZwBzUf_isx69jWeGsRoozgvBTRKtBPbYKQwIGAA6tHSQIgzvxCxdthQyi1WZHvXthPnJFkvyIKdCxYklRWgS55JNyabnMOjirZvvp1f0run5Fzxke3HRGeTGDX5Iu4jnpsMy7EgboZeszqLHn4UFfifGUs04WP6b10q7g9ReaHSDlu9upSnp1iu5zmovSIqsYNSpsR5WrN39tcxonOkXO91dBWDWWZcvN08oa1o2HoO) since it is open source. In this part of the assignment, we describe general instructions, however we will tailor the instructions for SFrame and scikit-learn.

- If you choose to use SFrame and scikit-learn, you should be able to follow the instructions here and complete the assessment. **All code samples given here will be applicable to SFrame and scikit-learn**.
- You are free to experiment with any tool of your choice, but **some many not produce correct numbers for the quiz questions.**

Load the Lending Club dataset

We will be using a dataset from the LendingClub (https://eventing.coursera.org/api/redirectStrict/Tpl43Y20w03sn-MaDAsrxr-8kMXqJSNqaa87UK7xiRSKOpkJhuDSM90RtbvtrJ5fXjofC7t1fkaWp6l40qjoyQ.MNxSq9x2wCRZ_sgobpw8nA.wZU2hL05nxTNkcBmka6OIyrMXoL9HQQH3LUhqzhhQtblRES4YGKhL5xYd6lR7nbKCD-

GEEpqhch2ziyyYBnMrxsIKP2f8nv7XKPqyHS5xn9xcbuSqpj2Ly-F8wPH1jVf78kKKBVc88XuqbMKUwjeF-NT2XXYJXrazFnqOhy9Ukbovnld8HJ6tmOLRAXhN9PfuK5u8TISefdcjaT1GSoDfOvfq94Ioufmz3SsRgwbj7vxB9MLKt0TtkPLJCGIJWFbUO4XHAoMr-15xyHGclExixdBhjxP6yv3mantSkHrzkA).

1. Load the dataset into a data frame named **loans**. Using SFrame, this would look like

```
import sframe
loans = sframe.SFrame('lending-club-data.gl/')
```

Note: To install SFrame (without installing GraphLab Create), run

```
pip install sframe
```

Exploring some features

2. Let's quickly explore what the dataset looks like. First, print out the column names to see what features we have in this dataset. On SFrame, you can run this code:

```
loans.column_names()
```

Here, we should see that we have some feature columns that have to do with grade of the loan, annual income, home ownership status, etc.

Exploring the target column

The target column (label column) of the dataset that we are interested in is called bad_loans. In this column **1** means a risky (bad) loan **0** means a safe loan.

In order to make this more intuitive and consistent with the lectures, we reassign the target to be:

- **+1** as a safe loan
- **-1** as a risky (bad) loan

3. We put this in a new column called **safe_loans**.

```
# safe_loans =  1 => safe
# safe_loans = -1 => risky
loans['safe_loans'] = loans['bad_loans'].apply(lambda x : +1 if x==0 else -1)
loans = loans.remove_column('bad_loans')
```

4. Now, let us explore the distribution of the column safe_loans. This gives us a sense of how many safe and risky loans are present in the dataset. Print out the percentage of safe loans and risky loans in the data frame.

You should have:

- Around 81% safe loans
- Around 19% risky loans

It looks like most of these loans are safe loans (thankfully). But this does make our problem of identifying risky loans challenging.

Features for the classification algorithm

5. In this assignment, we will be using a subset of features (categorical and numeric). The features we will be using are **described in the code comments** below. If you are a finance geek, the LendingClub (https://eventing.coursera.org/api/redirectStrict/Tpl43Y20w03sn-MaDAsrxr-8kMXqJSNqaa87UK7xiRSKOpkJhuDSM90RtbvtrJ5fXjofC7t1fkaWp6l40qjoyQ.MNxSq9x2wCRZ_sgobpw8nA.wZU2hL05nxTNkcBmka6OIyrMXoL9HQQH3LUhqzhhQtblRES4YGKhL5xYd6lR7nbKCD-GEEpqhch2ziyyYBnMrxsIKP2f8nv7XKPqyHS5xn9xcbuSqpj2Ly-F8wPH1jVf78kKKBVc88XuqbMKUwjeF-NT2XXYJXrazFnqOhy9UkbovnId8HJ6tmOLRAXhN9PfuK5u8TISefdcjaT1GSoDfOvfq94Ioufmz3SsRgwbj7vxB9MLKt0TtkPLJCGIJWFbUO4XHAoMr-15xyHGclExixdBhjxP6yv3mantSkHrzkA) website has a lot more details about these features. Extract these feature columns and target column from the dataset. We will only use these features.

```
features = ['grade',                     # grade of the loan
            'sub_grade',                 # sub-grade of the loan
            'short_emp',                 # one year or less of employment
            'emp_length_num',            # number of years of employment
            'home_ownership',            # home_ownership status: own, mortgage or rent
            'dti',                       # debt to income ratio
            'purpose',                   # the purpose of the loan
            'term',                      # the term of the loan
            'last_delinq_none',          # has borrower had a delinquincy
            'last_major_derog_none',     # has borrower had 90 day or worse rating
            'revol_util',                # percent of available credit being used
            'total_rec_late_fee',        # total late fees received to day
           ]

target = 'safe_loans'                    # prediction target (y) (+1 means safe, -1 is risky)

# Extract the feature columns and target column
loans = loans[features + [target]]
```

What remains now is a **subset of features** and the **target** that we will use for the rest of this notebook.

Sample data to balance classes

6. As we explored above, our data is disproportionally full of safe loans. Let's create two datasets: one with just the safe loans (**safe_loans_raw**) and one with just the risky loans (**risky_loans_raw**).

```
safe_loans_raw = loans[loans[target] == +1]
risky_loans_raw = loans[loans[target] == -1]
print "Number of safe loans  : %s" % len(safe_loans_raw)
print "Number of risky loans : %s" % len(risky_loans_raw)
```

One way to combat class imbalance is to undersample the larger class until the class distribution is approximately half and half. Here, we will undersample the larger class (safe loans) in order to balance out our dataset. This means we are throwing away many data points. We used seed=1 so everyone gets the same results.

```
# Since there are fewer risky loans than safe loans, find the ratio of the sizes
# and use that percentage to undersample the safe loans.
percentage = len(risky_loans_raw)/float(len(safe_loans_raw))

risky_loans = risky_loans_raw
safe_loans = safe_loans_raw.sample(percentage, seed=1)

# Append the risky_loans with the downsampled version of safe_loans
loans_data = risky_loans.append(safe_loans)
```

You can verify now that **loans_data** is comprised of approximately 50% safe loans and 50% risky loans.

Note: There are many approaches for dealing with imbalanced data, including some where we modify the learning algorithm. These approaches are beyond the scope of this course, but some of them are reviewed in this paper (https://eventing.coursera.org/api/redirectStrict/cbyl8-1r7-OlUoaNlS96u1dGzUcOjF_miGNBxaBHj9HYyG-vNTJybx93lbVuBAsi8VF2MsewERWnB2p7XD91ig.eFgO90qBAVfmK9o0pyqrlg.92mSjwRk-jEoO97md9zNcLFduwuBqemEohq03B8cVK0uGznpoxHZH7THPy19Z-moSTUF20CF3qBNPcsKELZq61F4yu_DNBPtS-e4fMC72bRSuXTSx3Kh8EG-oalrkPr4tK8AeKkPhGtvA7pmKy2_vQeyrjTVRy2zIQobVImu1OZFUVopwDRUdYoTgo8s3b-t1KMwhRot23WxBJl73AV9YwO6sZHVuQAUCgOjnoad4z_NDjEV1B0qwip18r1PG4pNnL7M2OH1uIUw4csQapUD_Iki1Chi5ErlnfUoKi0jgEdIHud68-1POsf0GeN9fH6uhNItv9aNJ-2aVfzRBe1OyutT9psgocTAUhqzsl7K39e6W35K8-Z0lCyGpC5IZhnrI5mVP8yPYnMXfI12fLgW4DMc9jP11cLfWoxWGUcybHI8s4ZkFVeNC6P2hVI-0zGM05QkZ3psK622nuPiIrVNzKi5ZOP-vTp813AE_8khgpzkA-Sp7w4kiXti_2I_kEafJVsr-UGdHOkIlOnIxP2P9m5oCC7edP13G9V65V0BOZBYu9WGYh57yj0fj2WDFJO0FJxEe7-n3VesZHQdOnUlI0ZdBJ93LINzZtwcOU9v4Y_pZsEnSV__5tm74_WN1sJs). For this assignment, we use the simplest possible approach, where we subsample the overly represented class to get a more balanced dataset. In general, and especially when the data is highly imbalanced, we recommend using more advanced methods.

One-hot encoding

7. For scikit-learn's decision tree implementation, it requires numerical values for it's data matrix. This means you will have to turn categorical variables into binary features via one-hot encoding. **The next assignment has more details about this.**

If you are using SFrame, feel free to use this piece of code as is. Refer to the SFrame API documentation for a deeper understanding. If you are using different machine learning software, make sure you prepare the data to be passed to the learning software.

```
loans_data = risky_loans.append(safe_loans)

categorical_variables = []
for feat_name, feat_type in zip(loans_data.column_names(), loans_data.column_types()):
    if feat_type == str:
        categorical_variables.append(feat_name)

for feature in categorical_variables:
    loans_data_one_hot_encoded = loans_data[feature].apply(lambda x: {x: 1})
    loans_data_unpacked = loans_data_one_hot_encoded.unpack(column_name_prefix=feature)

    # Change None's to 0's
    for column in loans_data_unpacked.column_names():
        loans_data_unpacked[column] = loans_data_unpacked[column].fillna(0)

    loans_data.remove_column(feature)
    loans_data.add_columns(loans_data_unpacked)
```

Split data into training and validation

8. We split the data into training and validation sets using an 80/20 split and specifying seed=1 so everyone gets the same results. Call the training and validation sets **train_data** and **validation_data**, respectively.

Note: In previous assignments, we have called this a **train-test split**. However, the portion of data that we don't train on will be used to help **select model parameters** (this is known as model selection). Thus, this portion of data should be called a **validation set**. Recall that examining performance of various potential models (i.e. models with different parameters) should be on validation set, while evaluation of the final selected model should always be on test data. Typically, we would also save a portion of the data (a real test set) to test our final model on or use cross-validation on the training set to select our final model. But for the learning purposes of this assignment, we won't do that.

```
train_data, validation_data = loans_data.random_split(.8, seed=1)
```

If you are not using SFrame, download the list of indices for the training and validation sets: module-5-assignment-1-train-idx.json (https://eventing.coursera.org/api/redirectStrict/egpOj4MQC1NBjeiJOf6hGqmIwGg1LaOJJR6JTQEYMhU05hG3w1CjfHJL8h9EfiUXUE2Gvz5IW3dUX7ls2ZOgGw.eHvbKUvsRv9hUXPT4y6-5A.q1uf7j96jvAUIu8LRztyWnbTLcpIYi7WpPe1z3OBbT2c_T9IKBX0ueW5cf0wkPyvguxmRxWXer-uxtjuJFWZAnx1mnIyRgS7b4jaPEmPIk0BUvRXtIo0T_az1i2HKD1PTKZ_Ju567d5wd3mGox76GWXKrNB0ZzEs0AbbzlLsDGJUaAkxWU8X5HvcVPC7FD0enIze4NXqp3MHyn0NiHqfKvwkHm9asIjMgstVjllvZrlkqSzq1TKlmJW4KUx_uEDk0heK1DwPsOsZXXT9zIGB2eyBRQ2DdS83Vr_Hj78Zz0YJFoIUqQbJfXq6urjWWR27NCL3ijOZSKbOpUufecJsbDugJ-2OvsFve7EYHtHzwDPnVxH0mQtbsZDLBvWFOwH1SCyQHVZsoR1S5PUKNywSOJoCxj7mxOvRiyid1e_kWzBTniU4nmVBjtp6Zs61hR_Wh8jamFpDdKe8FHzv7VSfP5BmQ4jqMc1xl37gjGgqg9pBbgP-SEjhfqpM2vwDOlGp), module-5-assignment-1-validation-idx.json (https://eventing.coursera.org/api/redirectStrict/4nRUaq0JIrHJJLNTb-Bt-LO28drtBbAgkqZh5uMytejDrOUjw0UsAsU7OCR71NQuyl2M6btwWp7RQ8WmtGGp3g.5gLrvhrkzOFy7m04CGxNGQ.g5TQ67mHYfHl1DuDVOXb6chTDKTIZHSd6bP1HUzkQIgicWYBPSQU3gnoII3Wx8Dhm312RKwZR4d_svnc7QtFrMQ_uUQl7ruqbxhz_xMKuaNdXrdJq4LJVhNMMRRSK-oTzwnJ9HP-fsuZhmkPbLgFxN63X1P_9U29t-cWrIEEJgRh2RqXp1V55ykMud1UlECVlID6dBNv_3ywiDgttlnUIgLaDFsG-JiIAZyp3YnYtu99Yf3tzzJcNKcL29S0Auv05Y9atvRz-hPmaC-6hpkTpFk0VF-oXFZ2NsVNYbgS6SjENmqJTFR_slTFHBP0g-3FE5sOzJpRG3I399RjZbG3n485XkxoPlpgf-uEDH-VqsGha9dsxMq5nqrZbMhRn3KNuGyb4XNs5olYqE_hsMrxR9uKlDHA-7F3d9EefSTx9f990xD1ILb1U_9_OxBK5LR47dW89hGP4DiaVD5Y0IXm7UPpWXyHWCWzSCaqwbHppnTtwMfDfBkdQll91mTUUm3fUcwwACoVCVIyOYJVrdnm4g). IMPORTANT: If you are using a programming language with 1-based indexing (e.g. R, Matlab), make sure to increment all indices by 1.

Build a decision tree classifier

9. Now, let's use the built-in scikit learn decision tree learner (sklearn.tree.DecisionTreeClassifier (https://eventing.coursera.org/api/redirectStrict/_Zn7gcnC1k-7Ko2zSS-vn81C2l52_o6oAzHGdiB32cCPjJRNuzfLkEvxRn_hPSN3uNEnjtppm_VNnFs0wqo7Eg.-1A5djxA0dAuXGmiYt2C-A.8s8rLuM9W4WJaQ9-7qMGRFCSflw8iK5pPEGsE7V-wrbp8QE0WFATznp7f5P7MqvtzBFXhmjfBV5PAWL8B_XEFoENQ16dveXX_TDY7FQpLBmAu3xF5S6GeQdD6AFxh5H5a7tpnXHGwGa8vd3JqXjes9BSWzBnWZhYAAF_qLYRvq_m5UkQuoiReFEOgRFBKIlqNlpNzrnNKNjtpgL-oi91Lupt7FmfL4egB0o0_YVvpx9IPMf22t8bLlXYMZDVbs6-vRvfvZxXBtdOukkutERqQ4YN7tyKgEBtvzhNi-qRTVSju2rRhwaWed0b94TfmpRop5kdkWNe9mNwhFUEGgyYSWoF5U75UDbVmotq7wSrGCzU4uIPgHmCr1ShHlBVu44GfyRO_58BkN1Ju4-FoHtyopjdtl9NmE2FYLK2nJD7faol1lmB1ZTTyKQV8pEcT8yU)) to create a loan prediction model on the training data. To do this, you will need to import **sklearn**, **sklearn.tree,** and **numpy**.

Note: You will have to first convert the SFrame into a numpy data matrix, and extract the target labels as a numpy array (Hint: you can use the **.to_numpy()** method call on SFrame to turn SFrames into numpy arrays). See the API (https://eventing.coursera.org/api/redirectStrict/_Zn7gcnC1k-7Ko2zSS-vn81C2l52_o6oAzHGdiB32cCPjJRNuzfLkEvxRn_hPSN3uNEnjtppm_VNnFs0wqo7Eg.-1A5djxA0dAuXGmiYt2C-A.8s8rLuM9W4WJaQ9-7qMGRFCSflw8iK5pPEGsE7V-wrbp8QE0WFATznp7f5P7MqvtzBFXhmjfBV5PAWL8B_XEFoENQ16dveXX_TDY7FQpLBmAu3xF5S6GeQdD6AFxh5H5a7tpnXHGwGa8vd3JqXjes9BSWzBnWZhYAAF_qLYRvq_m5UkQuoiReFEOgRFBKIlqNlpNzrnNKNjtpgL-oi91Lupt7FmfL4egB0o0_YVvpx9IPMf22t8bLlXYMZDVbs6-vRvfvZxXBtdOukkutERqQ4YN7tyKgEBtvzhNi-qRTVSju2rRhwaWed0b94TfmpRop5kdkWNe9mNwhFUEGgyYSWoF5U75UDbVmotq7wSrGCzU4uIPgHmCr1ShHlBVu44GfyRO_58BkN1Ju4-FoHtyopjdtl9NmE2FYLK2nJD7faol1lmB1ZTTyKQV8pEcT8yU) for more information. **Make sure to set max_depth=6**.

Call this model **decision_tree_model**.

10. Also train a tree using with **max_depth=2**. Call this model **small_model**.

Visualizing a learned model (Optional)

10a. For this optional section, we would like to see what the small learned tree looks like. If you are using scikit-learn and have the package Graphviz (https://eventing.coursera.org/api/redirectStrict/x_x0T_pssPZgNxLszEMK1XIywracE0yZ5Ay3lrqgkZF7Ana9Pno5PIB1

OX8PshOJonO4oEXiKaLS7KAFLzOiSQ.K3f3w9odDaehSpbj_eRkZQ.-0lrp5xj5lfB_q2SjGxRd54dirNAMmz7Gu4WVllrSsIE6UdaxZkYLLyl0H_EKSc6fu5kMxriezUoJCaXZ4OIpRVX-rJD8Hz1kGTaWwam7myeSZcQyR-0otbpZGjIp1hQMz826VsI8rFa28OJk-YkqjOxeLhXvWzNWOfoDECKc35AuyyMPwitPULlOeyKnOFkeuuZPHqBDSt5l1x4CM5Juu2QaBcWz_3opX0OOtvbGSMz04O_93MOBQQZ4pLYPe5iIA-abeJPcisW9TxLy2NgRG2blbSFrcA1RZoU6nyggykKIYLH9nk1HKTq-O__d2pjGZiCVqc6WVyICe1HZTuKeg), then you will be able to perform this section. If you are using a different software, try your best to follow along.

Visualize **small_model** in the software of your choice.

Making predictions

Let's consider two positive and two negative examples **from the validation set** and see what the model predicts. We will do the following:

- Predict whether or not a loan is safe.
- Predict the probability that a loan is safe.

11. First, let's grab 2 positive examples and 2 negative examples. In SFrame, that would be:

```
validation_safe_loans = validation_data[validation_data[target] == 1]
validation_risky_loans = validation_data[validation_data[target] == -1]

sample_validation_data_risky = validation_risky_loans[0:2]
sample_validation_data_safe = validation_safe_loans[0:2]

sample_validation_data = sample_validation_data_safe.append(sample_validation_data_risky)
sample_validation_data
```

12. Now, we will use our model to predict whether or not a loan is likely to default. For each row in the **sample_validation_data**, use the **decision_tree_model** to predict whether or not the loan is classified as a **safe loan**. (Hint: if you are using scikit-learn, you can use the **.predict()** method)

Quiz Question: What percentage of the predictions on **sample_validation_data** did **decision_tree_model** get correct?

Explore probability predictions

13. For each row in the **sample_validation_data**, what is the probability (according **decision_tree_model**) of a loan being classified as **safe**? (Hint: if you are using scikit-learn, you can use the **.predict_proba()** method)

Quiz Question: Which loan has the highest probability of being classified as a **safe loan**?

Checkpoint: Can you verify that for all the predictions with probability >= 0.5, the model predicted the label **+1**?

Tricky predictions!

14. Now, we will explore something pretty interesting. For each row in the **sample_validation_data**, what is the probability (according to **small_model**) of a loan being classified as **safe**?

Quiz Question: Notice that the probability preditions are the **exact same** for the 2nd and 3rd loans. Why would this happen?

Visualize the prediction on a tree

14a. Note that you should be able to look at the small tree (of depth 2), traverse it yourself, and visualize the prediction being made. Consider the following point in the **sample_validation_data:**

```
sample_validation_data[1]
```

If you have Graphviz, go ahead and re-visualize **small_model** here to do the traversing for this data point.

Quiz Question: Based on the visualized tree, what prediction would you make for this data point (according to **small_model**)? (If you don't have Graphviz, you can answer this quiz question by executing the next part.)

15. Now, verify your prediction by examining the prediction made using **small_model.**

Evaluating accuracy of the decision tree model

Recall that the accuracy is defined as follows:

$$\text{accuracy} = \frac{\#\text{ correctly classified examples}}{\#\text{ total examples}}$$

16. Evaluate the accuracy of **small_model** and **decision_tree_model** on the training data. (Hint: if you are using scikit-learn, you can use the **.score()** method)

Checkpoint: You should see that the **small_model** performs worse than the **decision_tree_model** on the training data.

17. Now, evaluate the accuracy of the **small_model** and **decision_tree_model** on the entire validation**_data**, not just the subsample considered above.

Quiz Question: What is the accuracy of decision_tree_model on the validation set, rounded to the nearest .01?

Evaluating accuracy of a complex decision tree model

Here, we will train a large decision tree with **max_depth**=10. This will allow the learned tree to become very deep, and result in a very complex model. Recall that in lecture, we prefer simpler models with similar predictive power. This will be an example of a more complicated model which has similar predictive power, i.e. something we don't want.

18. Using sklearn.tree.DecisionTreeClassifier (https://eventing.coursera.org/api/redirectStrict/ZS81doibh97jO93ciDjIlDX4ZnN6Z2wdW3z3zo3b6S7kPYNH3wNXW58NRrvRdXiHc09Z0f6MP230jXJW1FsFXg.d5GWmuIdEiafhEIrIKNT-A.6r_E0OSBonIcn4Ngs-uZ7NTavJI30b-eqVtzQ96qc5JZbWRlYOmsb4sJhQbwQ3A9Hh7J8db6ITVMqkO4GGezubzuyzQHkkiJe9mrmJTutIwc0RggNdjwiku-I1rIYFFQnMGJAgsNLUXnRNorbQRNI1H8FjaKUJVsGg_PpP6TZwag7OTTFMCfZip0z3tlZCkukp-JcCx14OTHa0cGVJ_Cx8seubopJ1e67RRv5HezhVZRNcYgQtMV1QQ8DkKYmjcVg9D2UDlaIMALQd7-t3WKc3N9uyWnr7t5XaBwWL-8GjxHg11jQGH6uI1jqdSULqhSjRKNuCA1CiaGrXTb2iHl08wAwAuWBky0yO9EwGZG1EGxK9CWrjdRRlBRIe1Q7IJc42Z_6aHA8F-6-hQ2Ig7aguZqsy2T8zQLogOWQ2BNkqJ-a9mRr91qHwsBqMTFDI0cOhydB3aNECsCVMBkgL8E8J36mWSOZpLZ-N7FUYQpBdBwAccqUt473hrg6ayDnu7kr8-Ks2ZbMgBbfuzC4SjMUCYe_IyMUc41xy4W0LRanNE), train a decision tree with maximum depth = 10. Call this model **big_model**.

19. Evaluate the accuracy of **big_model** on the training set and validation set.

Checkpoint: We should see that **big_model** has even better performance on the training set than **decision_tree_model** did on the training set.

Quiz Question: How does the performance of **big_model** on the validation set compare to **decision_tree_model** on the validation set? Is this a sign of overfitting?

Quantifying the cost of mistakes

Every mistake the model makes costs money. In this section, we will try and quantify the cost each mistake made by the model. Assume the following:

- **False negatives**: Loans that were actually safe but were predicted to be risky. This results in an oppurtunity cost of loosing a loan that would have otherwise been accepted.
- **False positives**: Loans that were actually risky but were predicted to be safe. These are much more expensive because it results in a risky loan being given.
- **Correct predictions**: All correct predictions don't typically incur any cost.

Let's write code that can compute the cost of mistakes made by the model. Complete the following 4 steps:

1. First, let us compute the predictions made by the model.
2. Second, compute the number of false positives.
3. Third, compute the number of false negatives.
4. Finally, compute the cost of mistakes made by the model by adding up the costs of true positives and false positves.

Quiz Question: Let's assume that each mistake costs us money: a false negative costs $10,000, while a false positive positive costs $20,000. What is the total cost of mistakes made by decision_tree_model on validation_data?