Mudcard

- **In addition to feature selection, are there certain situations when Lasso or Ridge regression is preferred?**
- **In what situations would we use ridge vs. lasso?**
- **If ridge regulation isn't the best for feature selection, then what do we use it for?**
- **I'm confused about which regularization option to pick - how would we know when to pick lasso or ridge?**
    - the model which optimizes test score is the preferred one
    - the main goal in ML is to minimize the generalization error so all similar questions are answered by this principle
    - you try every models you can and check which one performs best
- **Since lasso regularization sets feature parameters to 0 as alpha increases, and we interpret these 0-features as having relatively less predictive power than features which still remain at a given alpha level, can we infer the size of a coefficient as an indicator of relative predictive power of a feature? Or, can it differ based on value scale or other factors?**
    - under certain conditions yes
    - see PS7 problem 1 for a more detailed explanation :)
- **I still have problems in understanding the lasso regression and ridge regression. Why did ws converged to 0 when alpha increased to 1?**
    - the cost function is such that the larger the alpha and the weights, the larger the cost function is
    - as a result, the cost function is minimized when ws are small
- **The connection between MSE and the bias-variance bias tradeoff**
    - MSE is just one evaluation metric and the bias-variance trade-off can be illustrated with any evaluation metric
- **under what conditions will we be using log functions?**
    - I'm not sure what you mean by log functions
    - come to the office hours or elaborate on Ed Discussion
- **Why is Ridge Regression not suitable for feature selection?**
    - lasso is good for feature selection because some ws become 0 as the alpha increases
    - ridge is not good because the ws will never exactly be 0 regardless of how large alpha is
- **Is alpha the only hyperparameter for logistic regression (if we want to regularize)?**

    - for ridge and lasso, yes.
    - there are several regularization techniques beyond those two and some of them have more than one hyperparameters
- **Muddiest part is after finding most optimal alpha, how do we apply it through code?**
    - check again the two quizzes we solved last lecture
    - both quizzes answered your question
- **How do we initially assign alpha (eg alpha = np.logspace(-7,0,29) or alpha = np.logspace(-10,0,51), etc)?**
    - that's a bit subjective, there is no one good solution
    - but there are some considerations:
        - alpha should span several orders of magnitudes and there should be at least 1-2 values per order of magnitude, the more the better but there is no need to go crazy (i.e., you don't need to try hundreds of values)
        - if the best alpha is at the edge of your range, you need to increase the range because an even better alpha might be outside of the range
- **What are the typical use cases for ridge vs lasso?**
    - they are more sophisticated versions of linear and logistic regression and they often perform better too

The supervised ML pipeline

The goal: Use the training data (X and y) to develop a model which can accurately predict the target variable (y_new') for previously unseen data (X_new).

1. Exploratory Data Analysis (EDA): you need to understand your data and verify that it doesn't contain errors

- do as much EDA as you can!

2. Split the data into different sets: most often the sets are train, validation, and test (or holdout)

- practitioners often make errors in this step!
- you can split the data randomly, based on groups, based on time, or any other non-standard way if necessary to answer your ML question

3. Preprocess the data: ML models only work if X and Y are numbers! Some ML models additionally require each feature to have 0 mean and 1 standard deviation (standardized features)

- often the original features you get contain strings (for example a gender feature would contain 'male', 'female', 'non-binary', 'unknown') which needs to transformed into

numbers
- often the features are not standardized (e.g., age is between 0 and 100) but it needs to be standardized

4. Choose an evaluation metric: depends on the priorities of the stakeholders

- often requires quite a bit of thinking and ethical considerations

5. Choose one or more ML techniques: it is highly recommended that you try multiple models

- start with simple models like linear or logistic regression
- try also more complex models like nearest neighbors, support vector machines, random forest, etc.

6. Tune the hyperparameters of your ML models (aka cross-validation)

- ML techniques have hyperparameters that you need to optimize to achieve best performance
- for each ML model, decide which parameters to tune and what values to try
- loop through each parameter combination
  - train one model for each parameter combination
  - evaluate how well the model performs on the validation set
- take the parameter combo that gives the best validation score
- evaluate that model on the test set to report how well the model is expected to perform on previously unseen data

7. Interpret your model: black boxes are often not useful

- check if your model uses features that make sense (excellent tool for debugging)
- often model predictions are not enough, you need to be able to explain how the model arrived to a particular prediction (e.g., in health care)

Supervised ML algorithms

By the end of this **week**, you will be able to

- Summarize how decision trees, random forests, and support vector machines work
- Describe how the predictions of these techniques behave in classification and regression
- Describe which hyper-parameters should be tuned

Which ML algorithm to try on your dataset?

- there is no algo that performs well under all conditions! - no free lunch theorem
- you need to try as many as you can to find the one that performs best

- other than predictive power, what else is important for you?
  - how the model behaves with respect to outliers?
  - does the prediction varies smoothly with the feature values?
  - can the model capture non-linear dependencies?
  - is the model easy to interpret for a human?

Goal for this week: fill out the table:

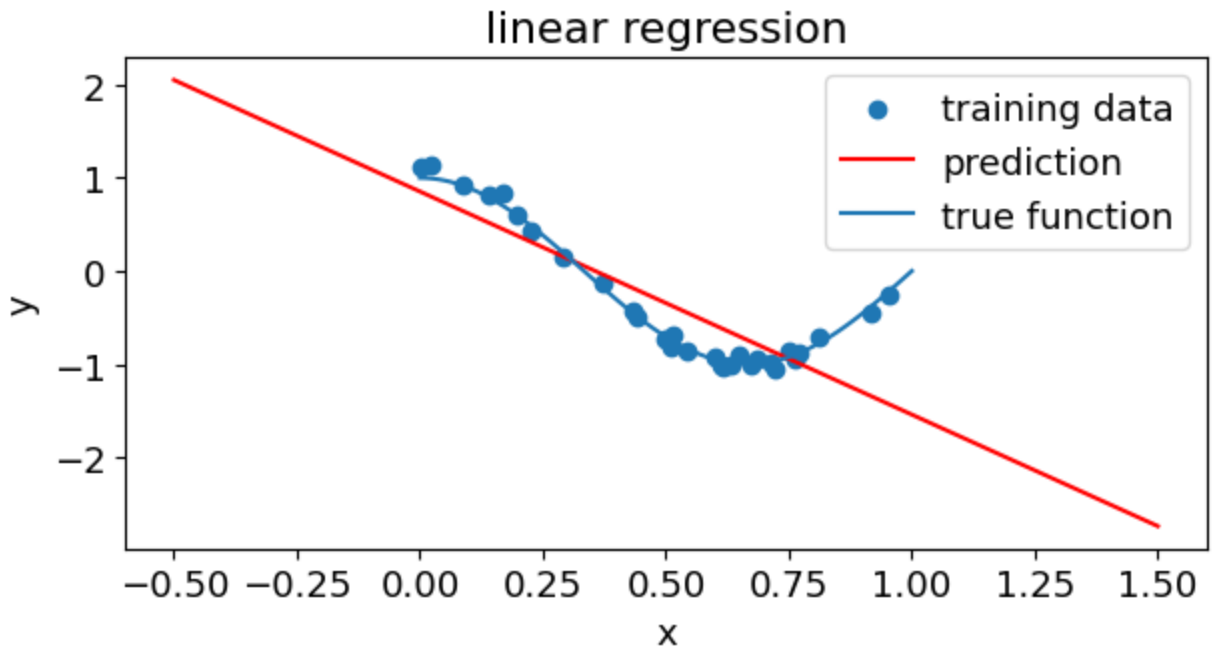
| ML algo | suitable for large datasets? | behaviour wrt outliers | non-linear? | params to tune |smooth predictions| easy to interpret?| |---------- |:----------: |:----------: |:----------: |:----------: |:----------: |:----------: | | linear regression | yes |tbd| no | l1 and/or l2 reg | yes | yes| | logistic regression | yes |tbd| no | l1 and/or l2 reg | yes | yes| | random forest regression | tbd | tbd | tbd | tbd | tbd | tbd| | random forest classification | tbd | tbd | tbd | tbd | tbd | tbd| | SVM rbf regression | tbd | tbd | tbd | tbd | tbd | tbd| | SVM rbf classification | tbd | tbd | tbd | tbd | tbd | tbd|

Linear regression

In [1]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
np.random.seed(10)
def true_fun(X):
    return np.cos(1.5 * np.pi * X)

n_samples = 30

X = np.random.rand(n_samples)
y = true_fun(X) + np.random.randn(n_samples) * 0.1

X_new = np.linspace(-0.5, 1.5, 2000)

reg = LinearRegression()
reg.fit(X[:, np.newaxis],y)
y_new = reg.predict(X_new[:, np.newaxis])
```

In [2]:
```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams.update({'font.size': 13})


plt.figure(figsize=(6.4,3.6))
plt.scatter(X,y,label='training data')
plt.plot(X_new,y_new,'r',label='prediction')
plt.plot(np.linspace(0, 1, 100),true_fun(np.linspace(0, 1, 100)),label='true fu
plt.xlabel('x')
plt.ylabel('y')
plt.title('linear regression')
plt.legend()
plt.tight_layout()
```

```
plt.savefig('figures/lin_reg.png',dpi=300)
plt.show()
```



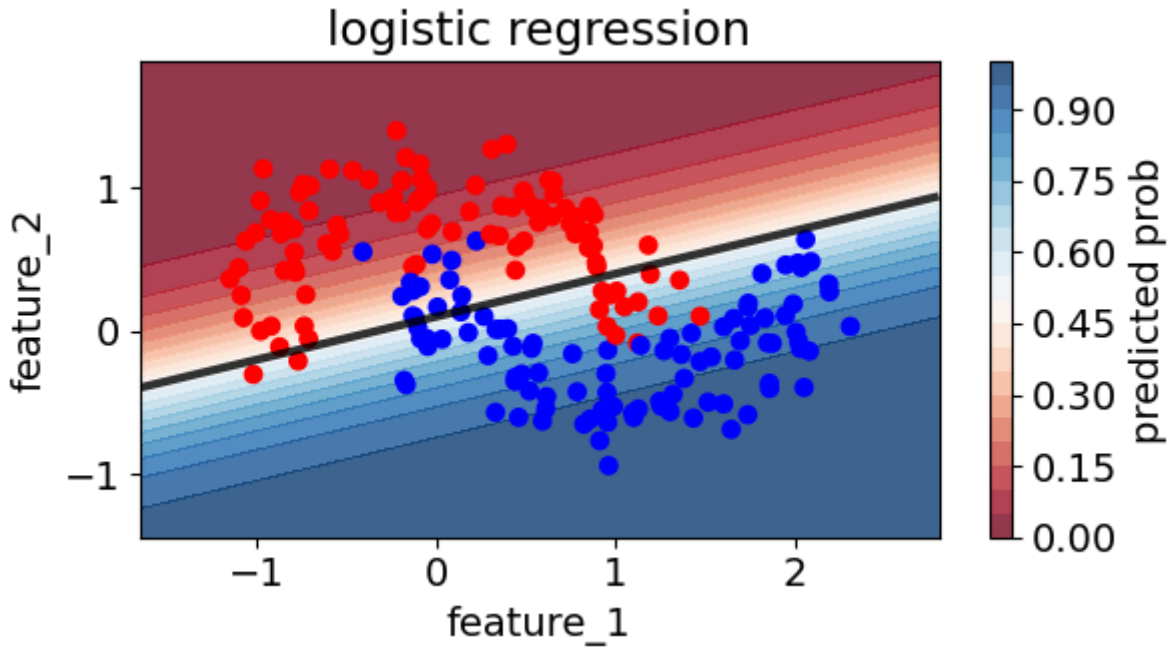
Logistic regression

In [3]:
```
from sklearn.datasets import make_moons
import numpy as np
from sklearn.linear_model import LogisticRegression
# create the data
X,y = make_moons(noise=0.2, random_state=1,n_samples=200)
# set the hyperparameters
clf = LogisticRegression()
# fit the model
clf.fit(X,y)
# predict new data
#y_new = clf.predict(X_new)
# predict probabilities
#y_new = clf.predict_proba(X_new)
```

Out[3]:
```
▾ LogisticRegression

LogisticRegression()
```

In [4]:
```
from matplotlib.colors import ListedColormap
from sklearn.preprocessing import StandardScaler
matplotlib.rcParams.update({'font.size': 14})

h = .02  # step size in the mesh

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
```

```
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
cm = plt.cm.RdBu

Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(6.4,3.6))
plt.contourf(xx, yy, Z, cmap=cm, alpha=.8,vmin=0,vmax=1,levels=np.arange(0,1.05
plt.colorbar(label='predicted prob')
plt.contour(xx, yy, Z, alpha=.8,vmin=0,vmax=1,levels=[0.5],colors=['k'],linewic
plt.scatter(X[:, 0], X[:, 1], c=y,cmap=cm_bright)
plt.xlabel('feature_1')
plt.ylabel('feature_2')
plt.title('logistic regression')
plt.tight_layout()
plt.savefig('figures/logistic_reg.png',dpi=300)
plt.show()
```



| ML algo | suitable for large datasets? | behaviour wrt outliers | non-linear? | params to tune |smooth predictions| easy to interpret?| |---------- |:----------: |:----------: |:----------: |:----------: |:----------: |:----------: | | linear regression | yes |linear extrapolation| no | l1 and/or l2 reg | yes | yes| | logistic regression | yes |scales with distance from the decision boundary | no | l1 and/or l2 reg | yes | yes| | random forest regression | tbd | tbd | tbd | tbd | tbd | tbd| | random forest classification | tbd | tbd | tbd | tbd | tbd | tbd| | SVM rbf regression | tbd | tbd | tbd | tbd | tbd | tbd| | SVM rbf classification | tbd | tbd | tbd | tbd | tbd | tbd|

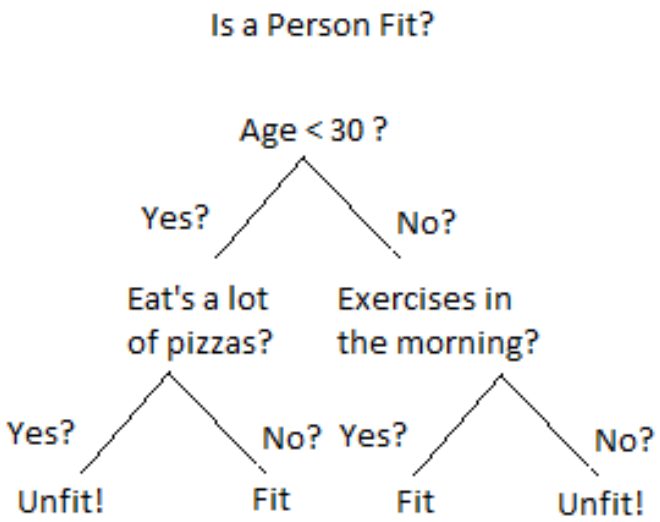
Quiz 1

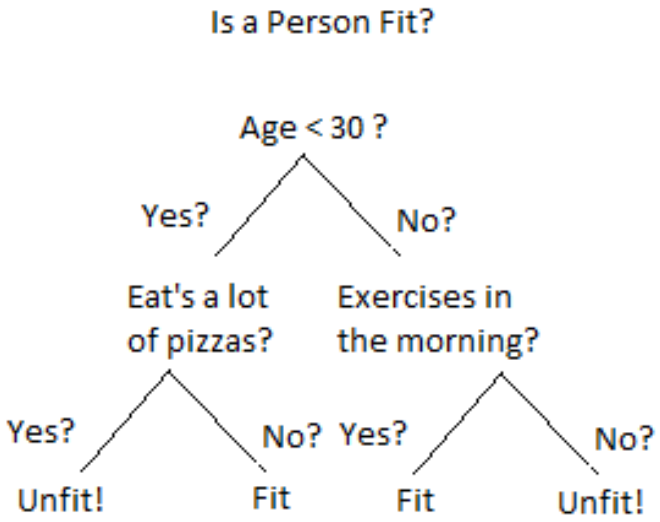
Calculate the predicted probabilities on X for the logistic regression model and determine what critical probability gives the best f1 score. Round to the second significant digit!

In [ ]:

Decision trees and random forests

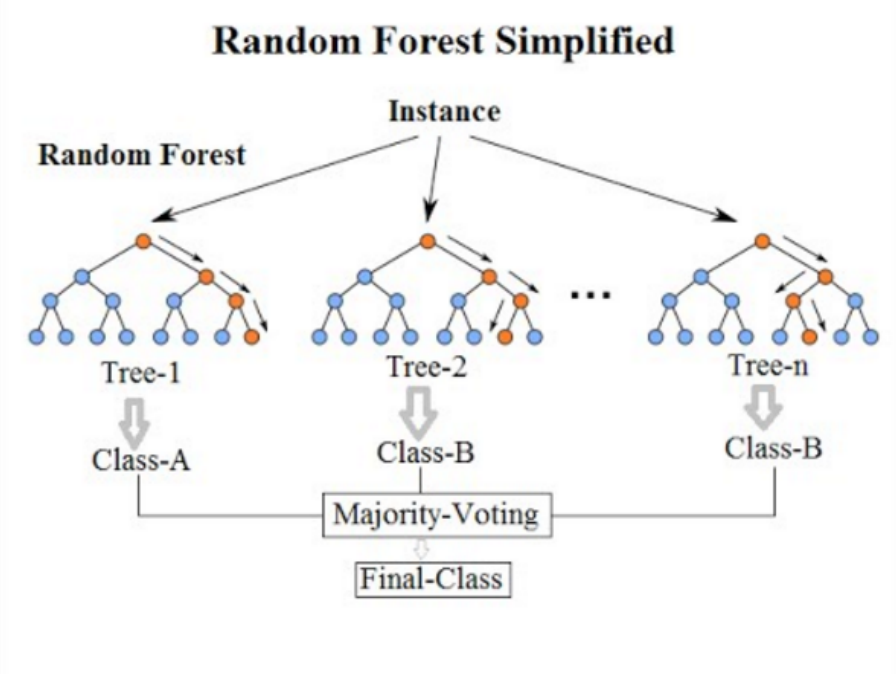
- Decision tree: the data is split according to certain features
- Here is an example tree fitted to data:





- Trees have nodes and leaves.
- The critical values and features in the nodes are determined automatically by minimizing a cost function.

- Random forest: ensemble of random decision trees
- Each tree sees a random subset of the training data, that's why the forest is random.



Quiz 2

- Use the dataset below and create a decision tree with max_depth = 2 to predict the target variable! What is your tree's prediction for each person?
- Remember, your tree does not need to predict everyone perfectly.
- It just needs to get as many people as possible right.

| X|age|gender (M=0, F=1)|is student?|is parent?|uses computer for work?|nr. of hours on c.|Like computer games?| |-|:-:|:-:|:-:|:-:|:-:|:-:|:-:|:-:| |**person 0**| 5|0|1|0|0|0.0|**1**| |**person 1**|48|1|0|1|0|1.8|**1**| |**person 2**|62|0|0|1|0|0.2|**0**| |**person 3**|10|1|1|0|0|2.4|**1**| |**person 4**|23|1|1|0|1|4.2|**0**| |**person 5**|36|0|0|0|1|3.1|**1**| |**person 6**|12|0|1|0|0|3.1|**1**| |**person 7**|85|0|0|0|1|1.0|**0**| |**person 8**|33|1|1|1|0|1.5|**0**| |**person 9**|56|0|0|0|1|0.1|**1**|

In [16]:
```
np.random.seed(1)
set(np.random.randint(10,size=10))
```

Out[16]:
```
{0, 1, 5, 6, 7, 8, 9}
```

Mud card

In [ ]:
```

```