Feature Engineering , Model Interpretation and intro to Hyperopt

EDVANCER EDUVENTURES

Agenda

Discussion Flow

- Notes on feature engineering
  - Revisit dummies for categorical data
  - Categorical embeddings
  - Coding cyclic time features
  - Revisit variable importance for feature selection
  - Transformed features
- Model Interpretation
  - Feature importance
  - Partial dependence plots
  - Local interpretation with LIME
  - Global interpretation with TREPAN
- Hyper parameter tuning with bayesian method (Hyperopt)
- Genetic Algorithm for another after model question

Feature Engineering

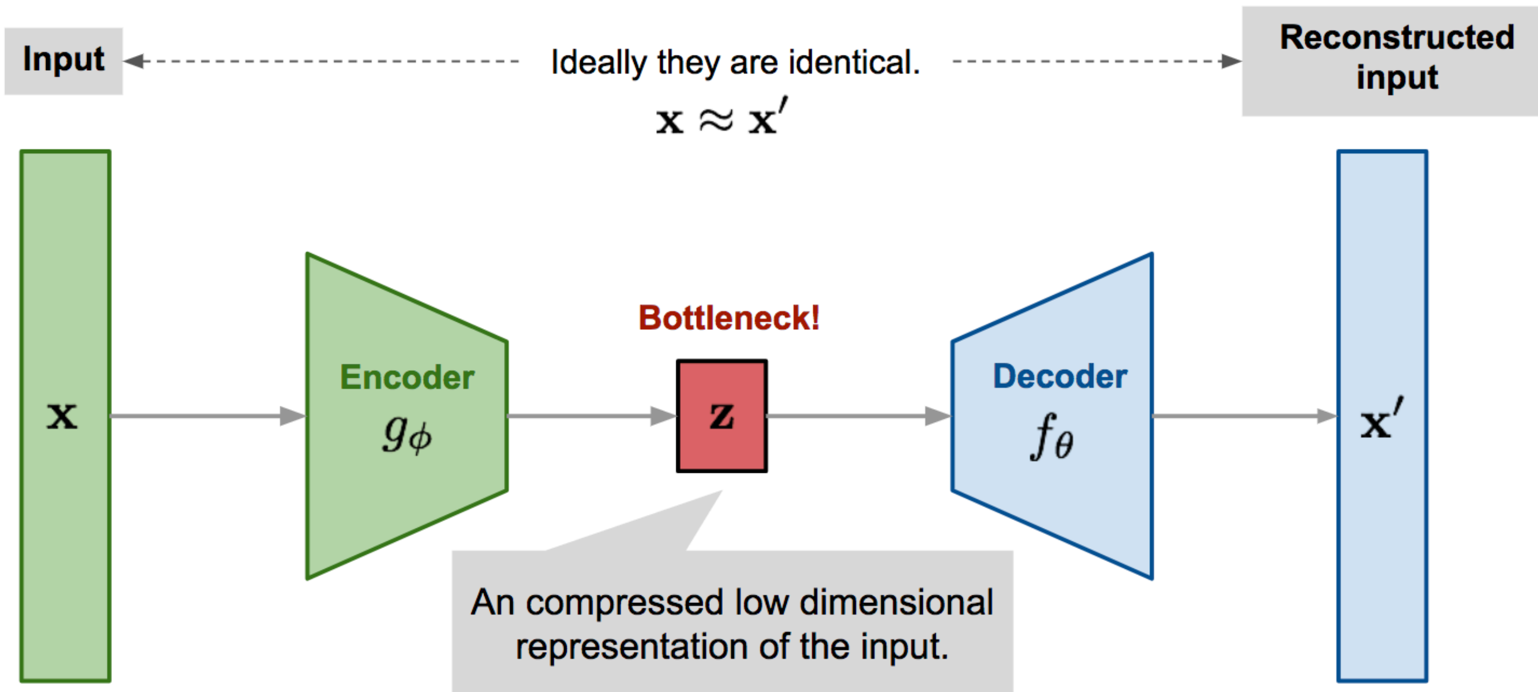
Creating Dummies for categorical data

- We create n-1 dummies for n categories
- However it doesn't make sense to create dummies for categories which have very few obs (why?)
- Potential Issues :
  - New data might have new categories
  - leads to data explosion
  - some practitioners use simple label encoding (not a good practice in industry from interpretation perspective)
  - Categorical var with high cardinality end up having more representation and algorithms selecting random set of features [RF,GBM etc] get biased towards them (why?, way out?)

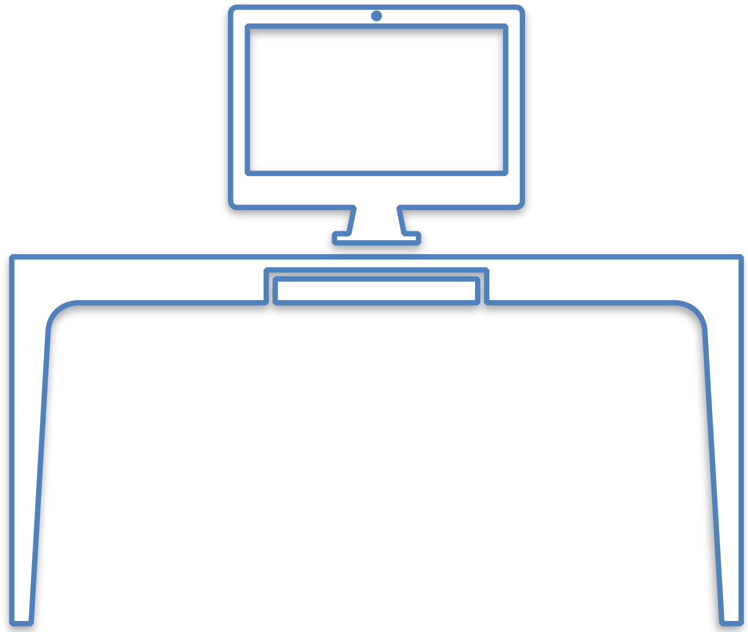
Categorical Embeddings

- Can be used for bringing down dimension of categorical representation
- We’ll be using auto encoders for the same (an idea from deep learning)
- Code etc will be new for you at this point.
- Focus on the idea and you can come back to it once having gone through deep learning course
- downside : embeddings as variables in the model will not be interpretable using whatever methods

Categorical Embeddings contd..



Lets see it in action in Python

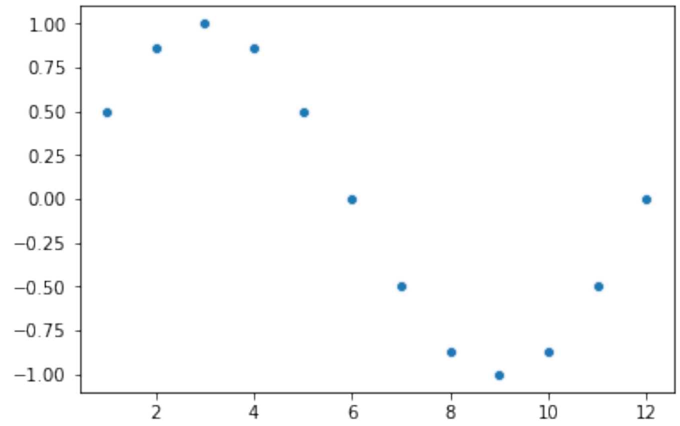


Coding Cyclic Time Features

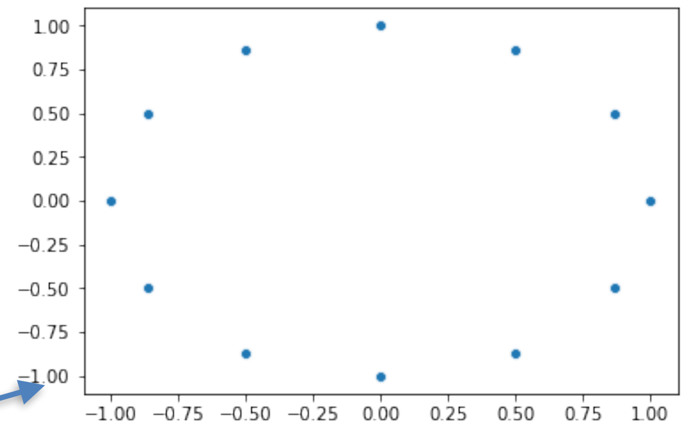
- Months , weekdays, hours etc .
- Simple numeric encoding doesn’t capture cyclic nature
- Dummies don’t capture the inherent order in them

encoding with pair of cyclic functions

Single cyclic function encoding

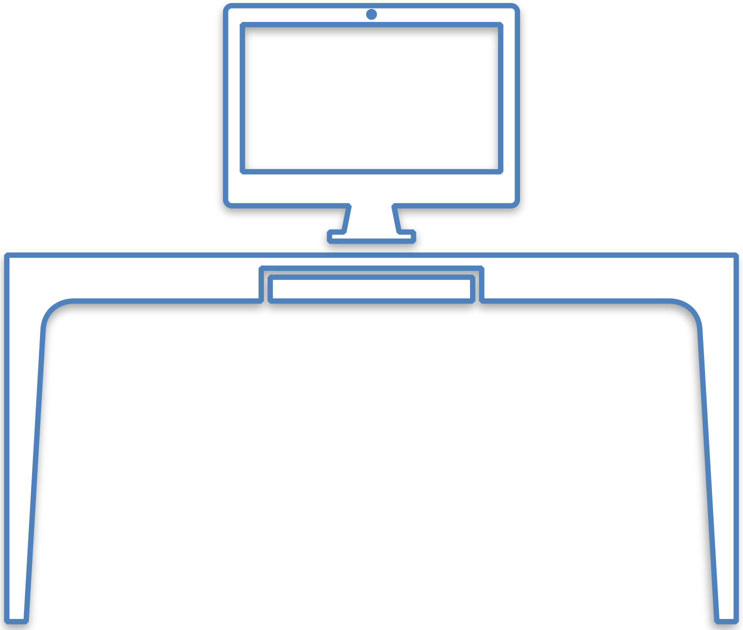


pair of cyclic function encoding



this is proper representation of cyclic features

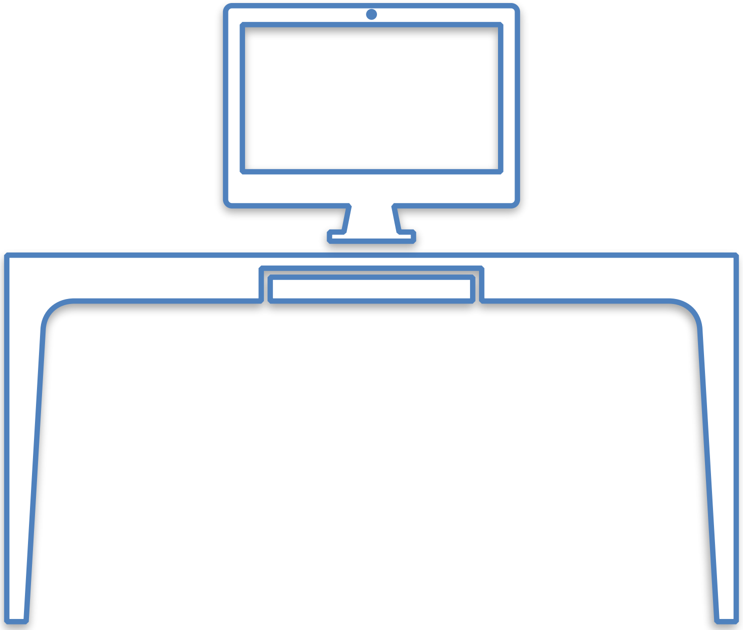
Lets see it in action in Python



Issues with feature importance in sklearn models

- Works well if all variables are dense numeric
- In such a scenario , adding a random column from a dense distribution and then comparing feature importance to discard vars works fine
- However :
  - discrete numeric columns have disadvantage of having less rules associated with them and run the risk of being wrongly labeled as less important than the dense random column
  - numeric columns might be at disadvantage at the step of random feature selection if there are high cardinality categorical features
  - traditional feature importance for feature selection will not be a great idea in such scenarios ; and these scenarios are pretty common
  - Package rfpimp implements permutation importance based on impact of absence of variables on model performance and thus more consistent across different kind of data
  - traditional feat imp will not always be unreliable (but why take chances)

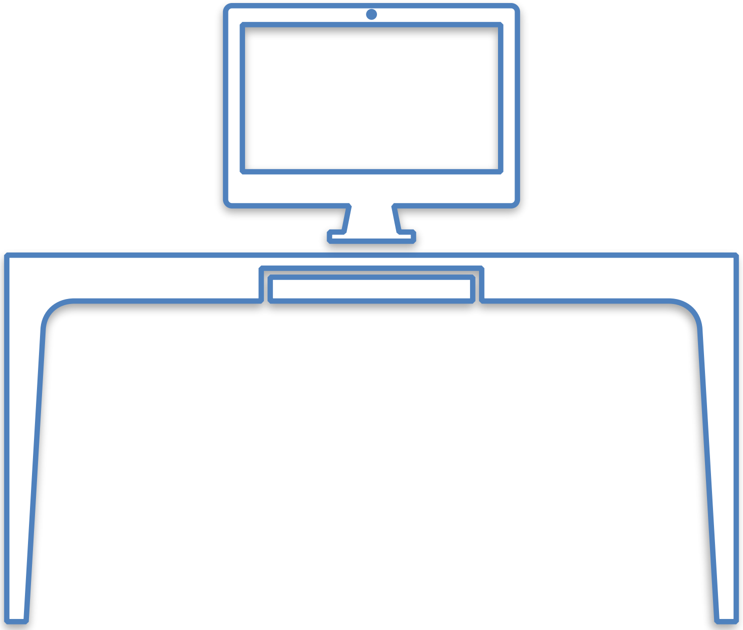
Lets see it in action in Python



Variable Transformation

- We use non-linear algos like Random Forest , Boosting Machines or SVM etc to model non-linear relationships
- So far we have assumed that these algorithms are equally good at extracting all kind of non-linear relationships [that's not true]
- We'll experiment and see that these algos [you can add more to the experiment in class] are pretty good at modelling polynomial , log etc kind of non-linear relationships
- However they don't perform well when target is related to inverse or ratios of variables
- This simply means that ratio vars and inverse are worthy transformations to try , not so much; many others which are something the algorithm can capture without us having to do feature engineering

Lets see it in action in Python

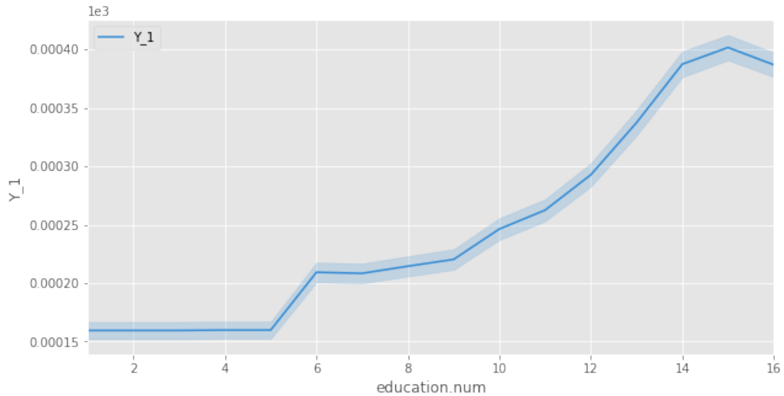


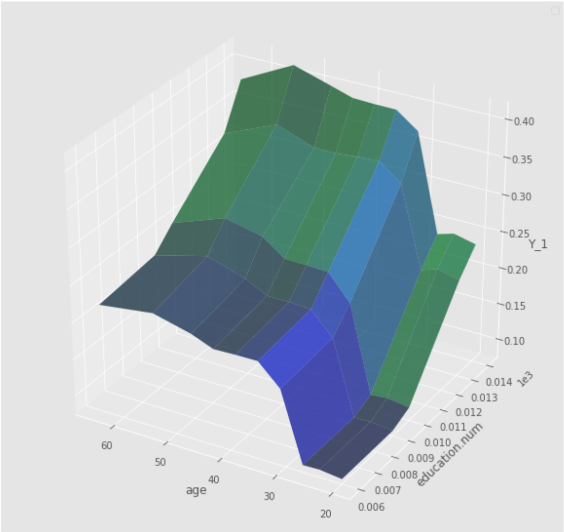
Model Interpretation
(for complex non-linear models)

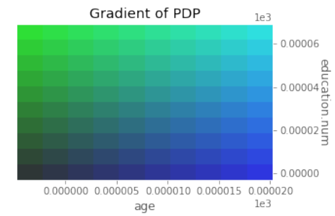
Global Interpretation

- Stands for assessing effect of any feature onto response across all training data instances , in other words the overall effect
- There are multiple ways to look at approximate estimate for the same
  - Feature Importance (discussed) : tells you which variables are important (doesn’t tell you nature of their impact)
  - Partial Dependence plots
  - Surrogate Models
- Package we’ll use : skater (other packages : eli5, shap (very slow for a good size data))

Partial dependence plots







- Considering all other features constant (average or sampled from an estimated distribution)
- Looks at how the model predictions vary with values of feature in question
- Good way to understand approximate non-linear relationship between features and response
- Actual pattern is further smoothed out for interpretability

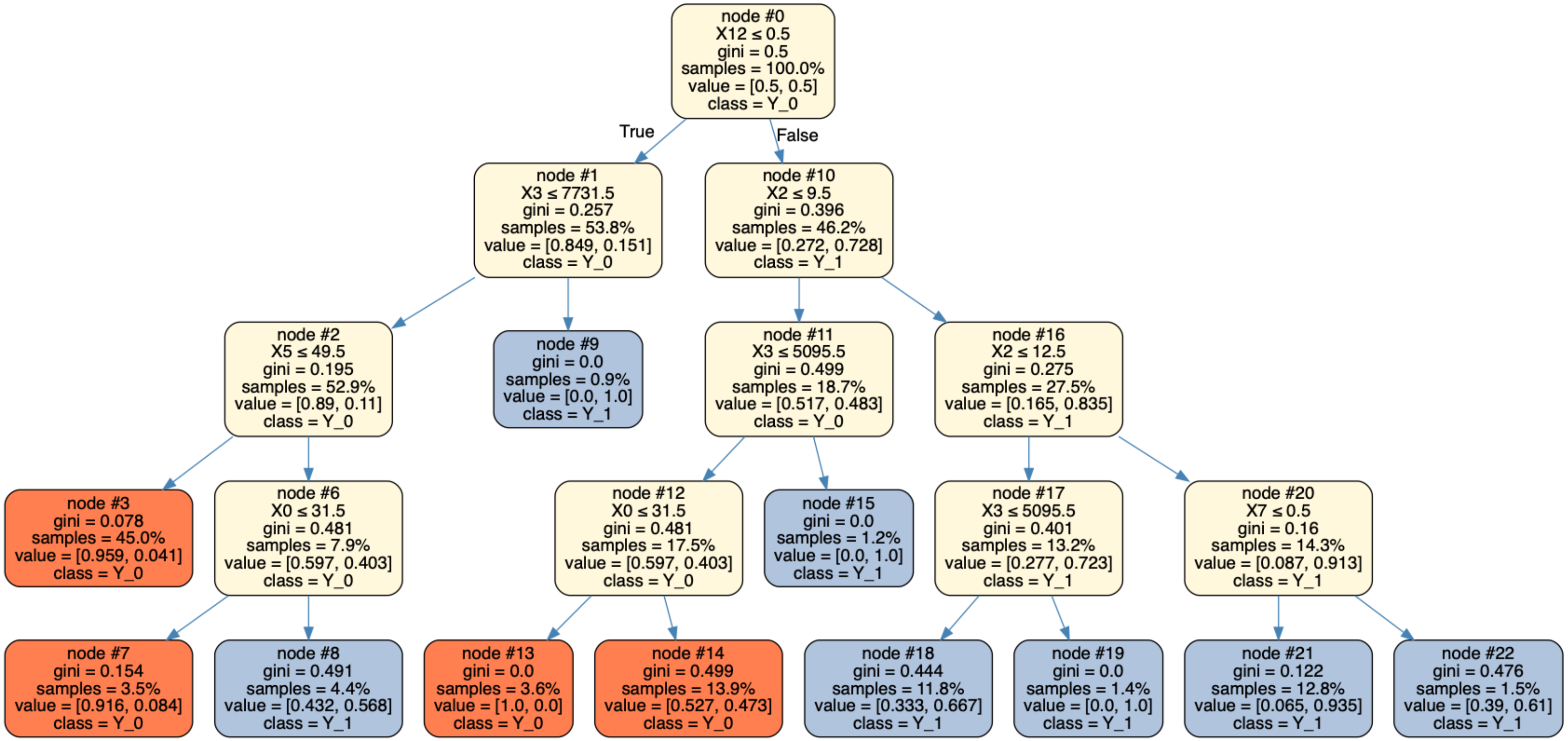
Surrogate model

- Idea : use an easy to interpret algorithm to build model considering predictions from complex model as target and original feature set
- Few pointers :
  - Its an interpretable model however its a (very) rough approximation of very complex relationship extracted by complex model
  - Most of the time surrogate models perform much worse than the actual model
  - Decision trees are popular choice for surrogate model (why not a linear model ?)
  - We can bring surrogate model (Dtree) close to complex model in performance by increasing complexity of Dtree (increasing depth) , however that compromises interpretability . Very deep trees are equally tough to interpret

TREPAN algo (used by skater for building surrogate Dtrees)

- Performs better than using traditional Dtrees as surrogate models
- How is it different from traditional Dtrees :
  - for building a tree , splits are chosen by making calls to original complex model (oracle)
  - split criterion : f(n) = reach(n)(1-fidelity(n)
  - reach(n) : number of obs reaching to node
  - fidelity(n) : accuracy w.r.t. to outcome of complex model (not the original outcome) [we are not trying to build a Dtree for original data , we are trying to approximate our complex model]

Example Surrogate Dtree



for : RandomForestClassifier(**{'criterion': 'entropy', 'max_depth': 14, 'max_features': 11, 'n_estimators': 320})

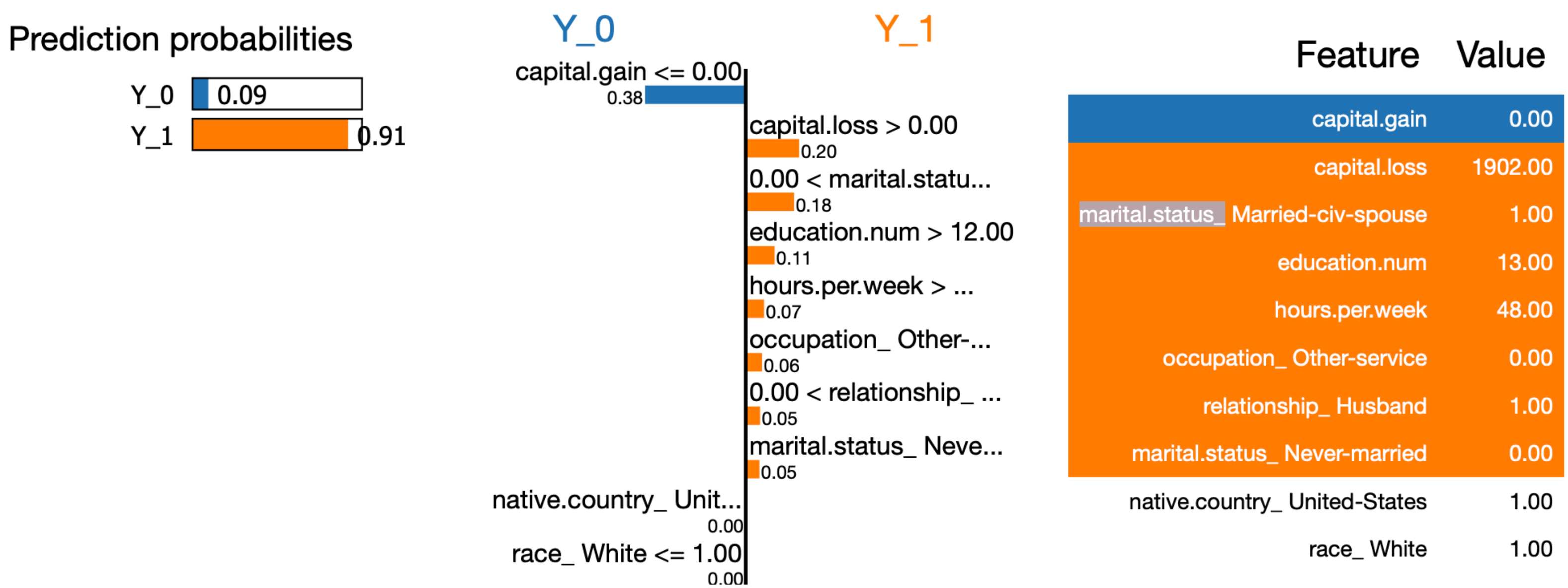
LIME (used by skater for local interpretation)

- Local interpretation ? : answering the question => why any single observation is being predicted to have a certain outcome ? what are the features which are contributing to that outcome and to what extent .
    - an example will be : say a particular application is denied a loan on the basis of our model . How do we answer , why that particular application was declined ?
- LIME : Local Interpretable Model-agnostic Explanations
- Notable alternative : shaply values (implemented in package shap with limiting downside of being very slow for a good size data [as of september 2019])
- Basic assumption : Every complex non-linear model is linear on a local scale

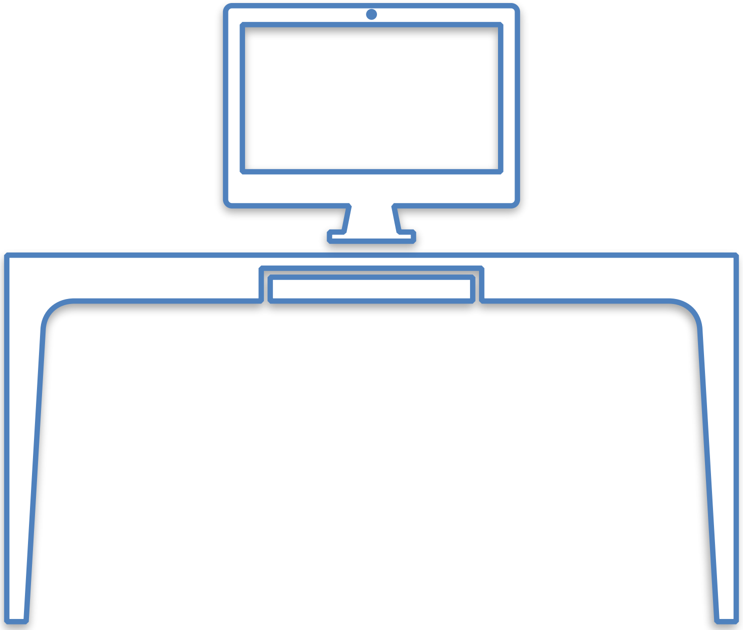
LIME algorithm sketch

- Take input the instance/observation and the complex model
- Create perturbed obs around the given instance [by sampling feature values from approximate distribution around the instance]
- use complex model to make predictions on perturbed data [this is used as target by local model , as we are trying to approximate this guy]
- Use simple Lasso model to select features with non-zero coefficient
- Build a simple linear model with weighted least square loss [taking only the features selected by Lasso in previous step. Here weight for each perturbed data point is defined as inverse of its distance from the actual instance]
- Coefficient and their signs are used as measure of impact of each of these features towards the outcome for instance in question

Example LIME Outcome



Lets see it in action in Python



Bayesian Hyper-parameter Tuning (package : hyper opt)

Motivation

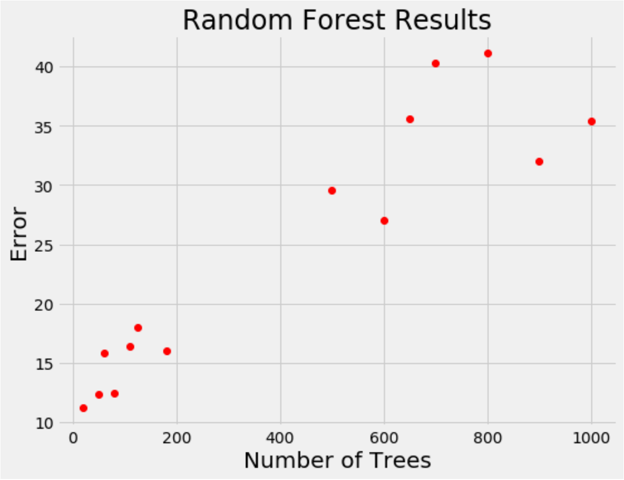
- Grid search becomes infeasible very soon with large parameter combinations to try
- Random search is better than grid search but is not efficient
- Why does random search work in first place ?

Why random search works ?

- Random search has a probability of 95% of finding a combination of parameters within the 5% optima with only 60 iterations. [How?]

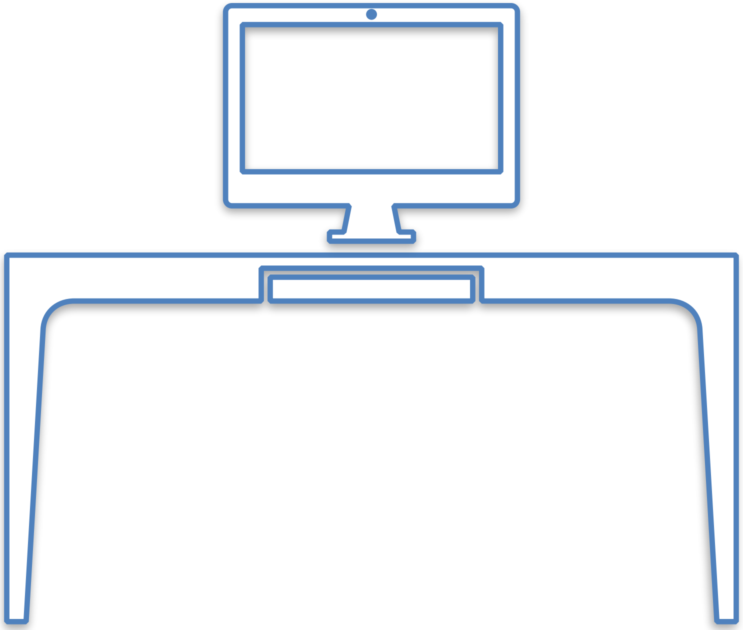
- Consider that there exist a true maxima and consider a 5% interval around that [as in if you randomly select combination, there is a 5% chance that it will land in that interval]
- If we select n random combinations , probability of all of them missing that 5% interval is $(1-0.05)^n$
- So the probability of at least one of them hitting the 5% interval will be $1-(1-0.05)^n$
- If we want 95% chance of at least one choice being in the 5% interval we just need to solve this for n : $1- (1-0.05)^n >0.95$
- This gives us n>=60
- We can increase the chances by using more iterations

Motivation contd.. : why is random search inefficient



- Lets say you tried some n random combinations of parameters and their performance looked like as given in the figure
- Clearly an intelligent choice for next value of number of trees to try will be near 100, as that interval seems to have lower error
- However random search doesn't consider this pattern
- Next parameter combination to be tried will be chosen with uniform probability from the entire search space
- Bayesian optimisation considers this distribution of performance across parameter combinations
- For first few iterations it simply does a random search
- Then it fits a distribution using TPE algorithm , which is more informative than a simple uniform probability distribution
- Next parameter combination is now sampled from the interval where performance is improving
- Distribution keeps on getting updated as we try more and more combinations
- It converges to optimal values of hyper parameter much faster than random search
- However , for a large number of hyper-parameters , performance is equivalent to random search

Lets see it in action in Python



Finding feature values for optimal target requirement

With Genetic Algorithm

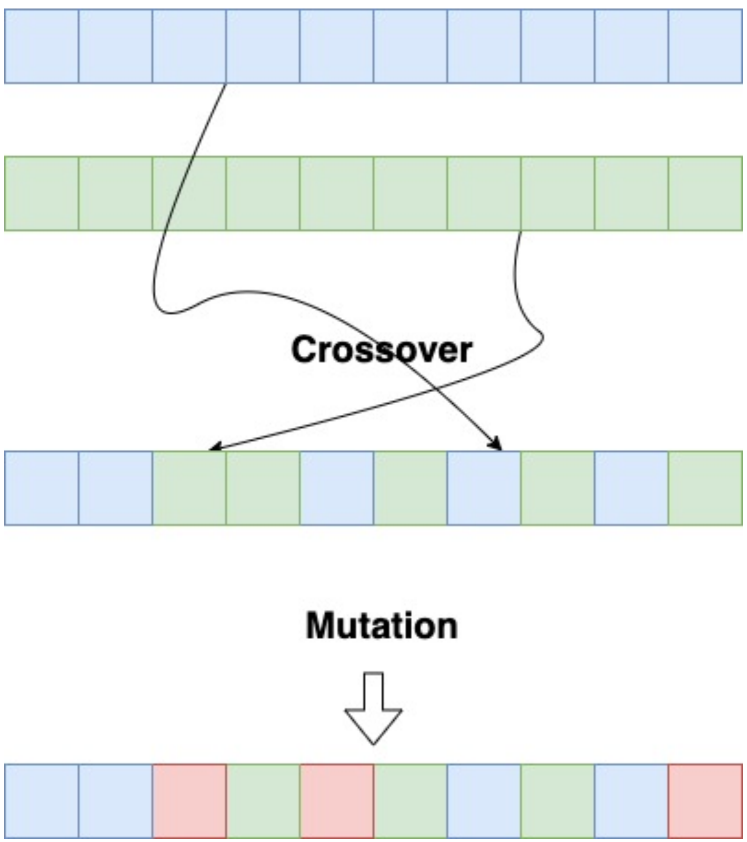
Example Problem

- Consider the interest rate prediction problem that we worked on
- The model that we built answers the question: what interest rate I will get, given the feature values of my loan application
- What if some stakeholder has this question : If their loan request is for $5000 then having what credit score will get them least interest rate.
- We dont simple linear model or some simple functional model which can be optimised for minimum target with given value of requested amount
- We'll make use of genetic algorithm to find the answer

Genetic Algorithm

- Genetic algorithm follows the idea of evolution at a broader level
- Assume that we chose the healthiest individuals to generate next generation [with crossover and mutation] and repeat this process with the generation that we get
- This will lead to better and better individuals in the population as we go from one generation to next
- Its possible that the starting generation [random values] , do not have good genes to start with, to counter that, each generation also receives some random values being added as part of the population; in order to provide a chance at discovering alternate better candidates from the general populace

How does it work



- Similarity to genetic evolution put aside, crossover lets the algorithm explore more distributions of values from ‘good’ parents [at the time of checking performance for selecting best, we check performance of individuals separately]
- Mutation allows algorithm to explore , so far possibly left out distribution space
- Multiple generations generated through this algorithm , eventually converges to optimal value

Lets see it in action in Python