WAYNE STATE UNIVERSITY

Model Assessment

Dr. Ratna Babu Chinnam

Industrial & Systems Engineering

Wayne State University

Motivation

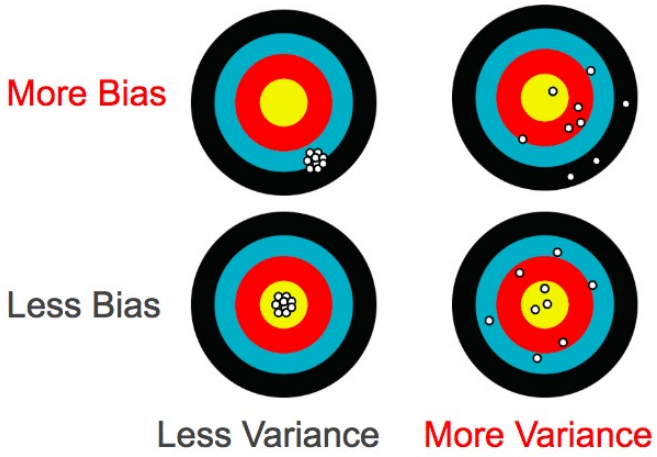
Google Says It Will Address AI, ML Model Bias with Technology called TCAV

By Larry Dignan | May 7, 2019 | ZDNet | Link

- Google CEO Sundar Pichai said the company is working to make its AI and ML models more transparent to defend against bias.
- Pichai outlined several AI enhancements, but the bigger takeaway for developers and data scientists may be something called TCAV.
- Testing with Concept Activation Vectors (TCAV) is an interpretability method to help us understand signals driving neural network predictions.
- In theory, TCAV's ability to understand signals could surface bias because it would highlight whether males were a signal over females and surface other issues such as race, income and location. Using TCAV, one can see how high value concepts are valued.

Judging Model Performance & Robustness

- Accuracy isn't Everything
  - Timeliness / Lead time to action
  - Class imbalances / Bias
  - Cost differences: Need cost-sensitive learning
- Performance Repeatability
  - Ideal conditions vs real-world
- Variable Selection/Influence Consistency
- False Alarms Fatigue!
- Purely Data-Driven vs Mechanistic/Physics Based Models
- Ensembles: Combine Approaches for Robustness



Real-life Examples of ML Discrimination

- **Racial Bias in US Health Care Risk Algorithm** (Vartan 2019 | Link)
  - Algorithm used on > 200M people in US hospitals to predict which patients would need extra medical care heavily favored white patients
    - Race itself wasn't a variable used in this algorithm, however, another variable highly correlated to race was, healthcare cost history.
    - For various reasons, black patients incurred lower health-care costs than white patients with the same conditions on average.
  - Optum helped reduce the level of bias by 80%.
- **AI Bias in US COMPAS Algorithm** (Larson et al., 2016 | Propublica)
  - Correctional Offender Management Profiling for Alternative Sanctions
  - Used in US courts to predict likelihood that a defendant becomes a recidivist
  - Model predicted twice as many false positives for black offenders (45%)
- **Amazon's Sexist Hiring Algorithm** (Dastin 2018 | Reuters)
  - Models trained to vet applicants by patterns in résumés from 10-years
  - System taught itself that male candidates were preferable.
    - It penalized résumés that included the word "women's", as in "women's chess club captain". And it downgraded graduates of two all-women's colleges.
  - Amazon edited the programs to make them neutral to these terms. But that was no guarantee that the machines would not devise other ways of sorting candidates that could prove discriminatory, the people said.
  - Company ultimately disbanded the team because executives lost hope

Source: Shin (2020) Real-life Examples of Discriminating AI| Link

Model Assessment

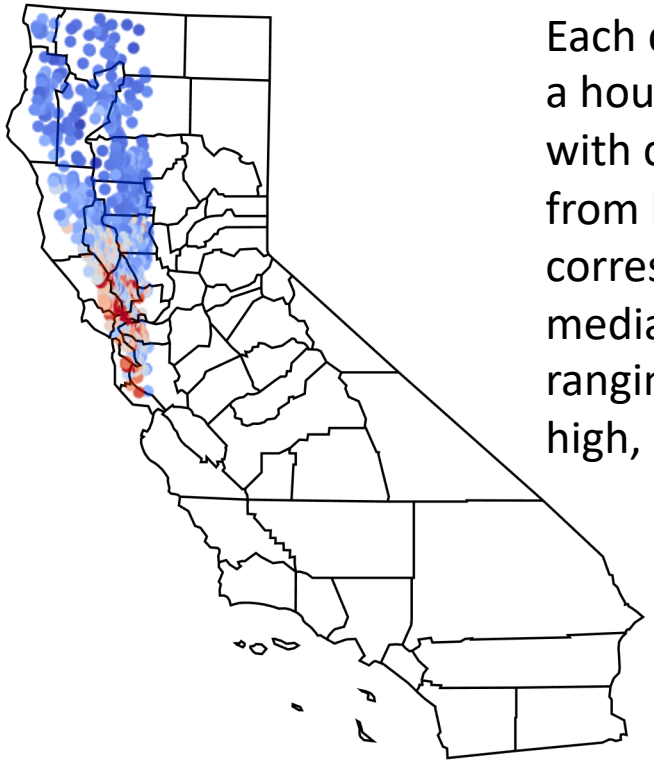
- First Requirement: Dataset offers (best) features to carry out task
- Feature selection/extraction might be necessary
- Small datasets: *K*-fold cross-validation strategy and/or regularization
- Build a model that best addresses the *bias-variance dilemma*
  - For “small” datasets, as one decreases bias, variance increases (and vice versa)
- Effective Model:
  - Selecting most appropriate modeling approach
  - Optimize model complexity (e.g., in MLPs, # layers, # nodes/layer), model parameters (e.g., transfer function)
  - Appropriate learning algorithm and parameters
- Vigorous Testing & Sensitivity Analysis

Some Sources of Model Bias

- Missing Feature Values
  - If your data set has features that have missing values for a large number of examples, that could be an indicator that certain key characteristics of your data set are under-represented.
- Unexpected Feature Values
- Data Skew
  - Geographical Bias Example: If this unrepresentative sample were used to train a model to predict California housing prices statewide, the lack of housing data from southern portions of California would be problematic.

	longitude	latitude	total_rooms	population	households	median_income	median_house_value
count	17000.0	17000.0	17000.0	3000.0	3000.0	3000.0	17000.0

California Housing Dataset



Each dot represents a housing block, with colors ranging from blue to red corresponding to median house price ranging from low to high, respectively.

Evaluating for Bias: Example

- Consider a model developed to predict presence of tumors
  - 500 records each from female and male patients
- Calculated metrics separately for female and male patients show stark differences

Confusion Matrix

True Positives (TPs): 16	False Positives (FPs): 4
False Negatives (FNs): 6	True Negatives (TNs): 974

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{16}{16 + 4} = 0.800$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{16}{16 + 6} = 0.727$$

Female Patient Results

True Positives (TPs): 10	False Positives (FPs): 1
False Negatives (FNs): 1	True Negatives (TNs): 48

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{10}{10 + 1} = 0.909$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{10}{10 + 1} = 0.909$$

Male Patient Results

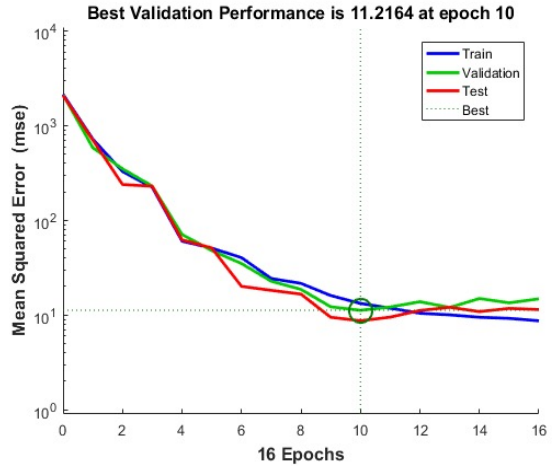
True Positives (TPs): 6	False Positives (FPs): 3
False Negatives (FNs): 5	True Negatives (TNs): 486

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{6}{6 + 3} = 0.667$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{6}{6 + 5} = 0.545$$

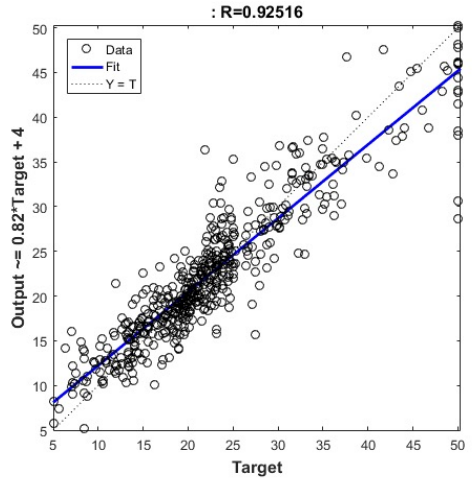
Function Approximation & Regression

- Performance criterion for learning model should be a “complete” measure
  - Accuracy and precision (e.g., MSE and MAD are complete measures but not $E(error)$)
- Model errors across training, validation, and testing datasets should be consistent
- Regression plots are useful
  - For training, validation, and testing datasets
  - Ideally have a high R^2 and an intercept of zero and slope of 1



Sample Cross-Validation MSE Plot

```
Matlab:-> [net,tr] = train(net,x,t); plotperform(tr)
```

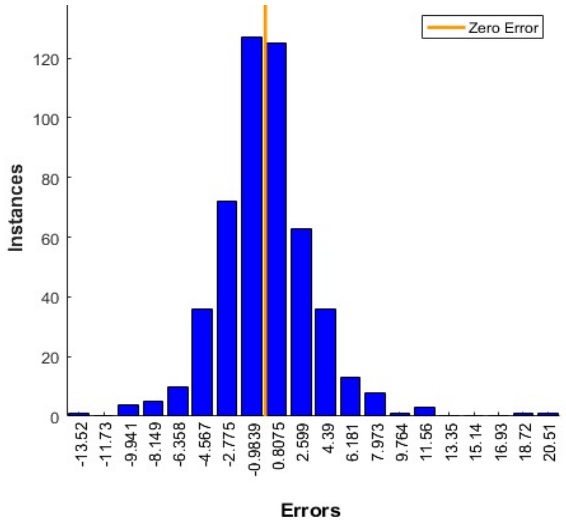


Sample Plot of Actual Vs. Predicted

```
Matlab:-> y = net(x); plotregression(t,y)
```

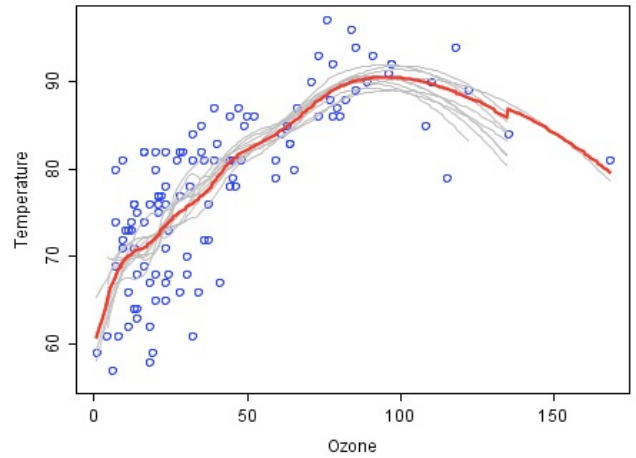
Function Approximation & Regression …

- Error histograms are also quite useful
  - Most errors should be close to zero
  - Significant outliers (large errors) should be investigated further
    - Data collection errors, missing variables
- Track performance across replications
  - Many machine learning models can get stuck in local optimum solutions
  - Training process must be repeated several times to demonstrate that results are consistent



Sample residual error histogram

```
Matlab:-> e = t - y; ploterrhist(e)
```

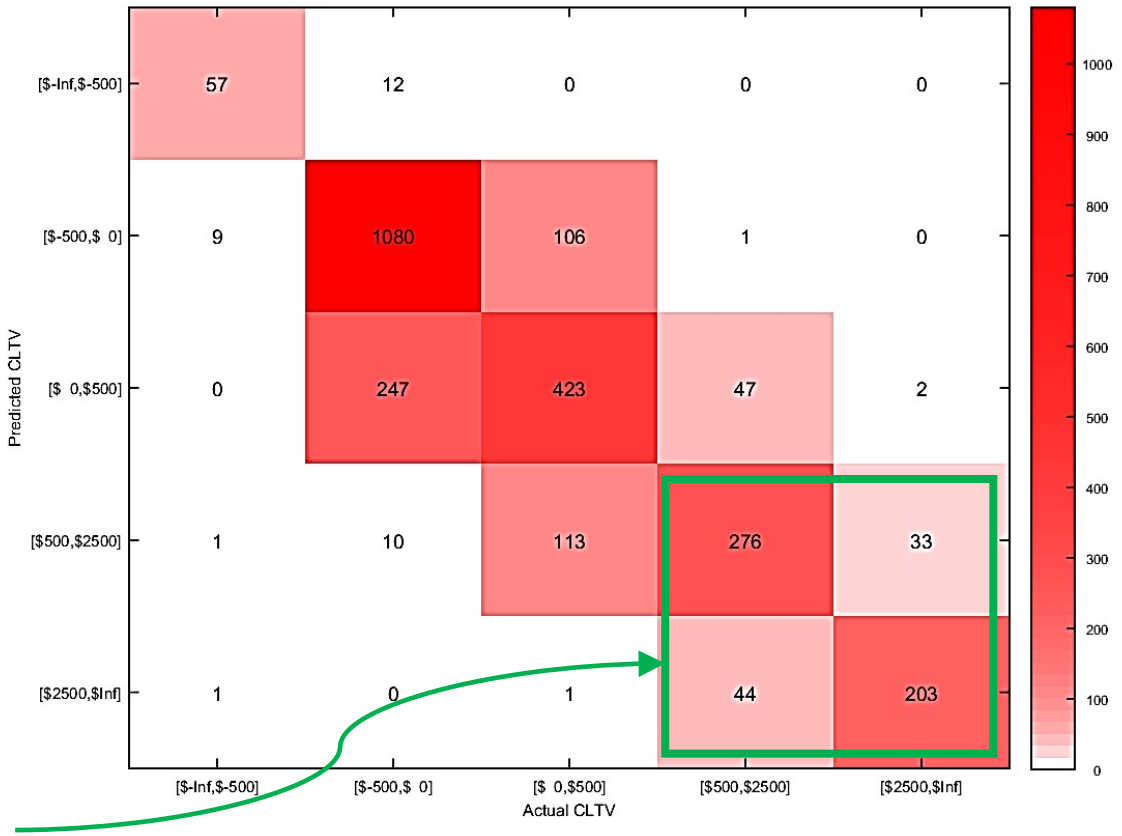


Results from different runs (gray lines)

```
Matlab:-> y = net(x); plotregression(t,y)
```

Function Approximation & Regression …

- Binning errors by range might be of interest for certain applications
  - Certain types of errors might be more important
  - Example: Table reports actual versus estimated Customer Life-Time Values (CLTV) from a model



Client's Interest

Sample matrix of predicted versus actual output

Pattern Recognition (Classifiers)

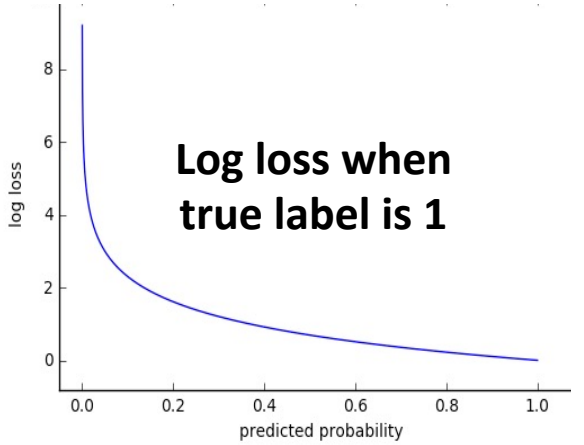
- **Cross-entropy loss** appropriate for training classifiers (Link1, Link2)
  - Penalizes extremely inaccurate outputs (p near $1-t$) with little penalty for fairly correct classifications (p near t)

$$\text{CE Loss} = -\sum_{c=1}^{M} t_{o,c} \log(p_{o,c})$$

M: number of classes

$t_{o,c}$: Binary target (0 or 1) if label c is correct for observation o

$p_{o,c}$: predicted probability observation o is of class c



- Example: Let us take a three-class classification problem
  - True Output (Target) for a particular observation: [0 0 1]
  - Predicted Output for this observation: [0.04 0.13 0.83]

$$\text{CE Loss} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

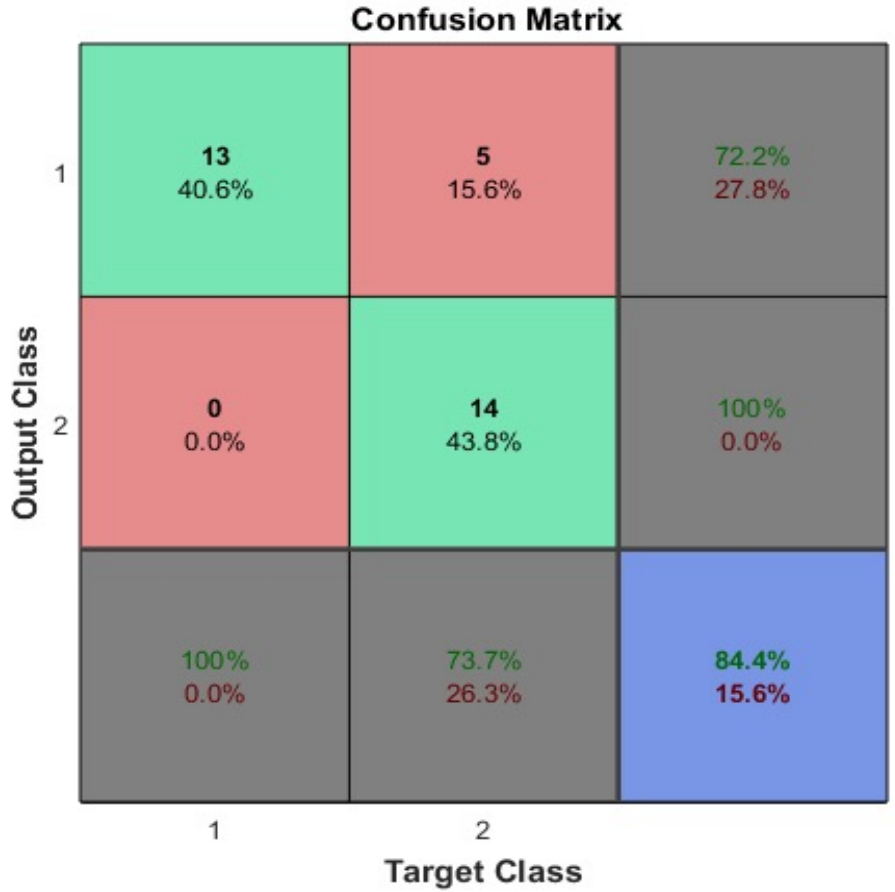
$$= -[0\times\log(0.04) + 0\times\log(0.13) + 1\times\log(0.83)] = -\log(0.83)$$

Distinguish multi-class classification from multi-label classification!

A sample can belong to more than 1 class.

Pattern Recognition (Classifiers): Confusion Matrix

- An effective performance measure is "***confusion matrix***"
  - Shows %s of correct and incorrect classifications
  - Diagonal entries are correct (green)



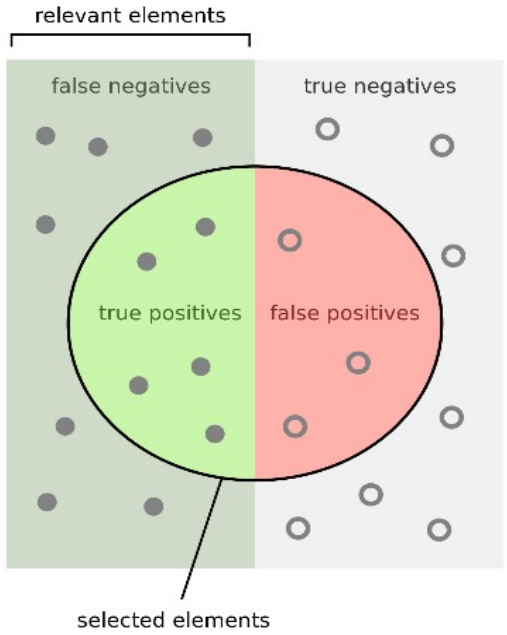
Sample Confusion Matrix for Testing Dataset With Two Target Classes

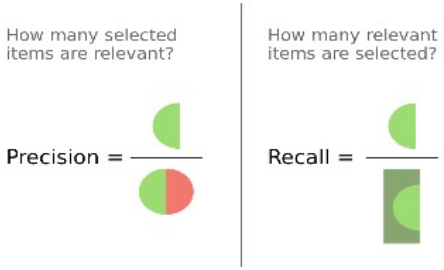
```
Matlab:-> [net,tr] = train(net,x,t);
testX = x(:,tr.testInd); testT = t(:,tr.testInd); <- Testing Observations
testY = net(testX); plotconfusion(testT,testY)
```

Pattern Recognition (Classifiers): Confusion Matrix

Binary Classification Example

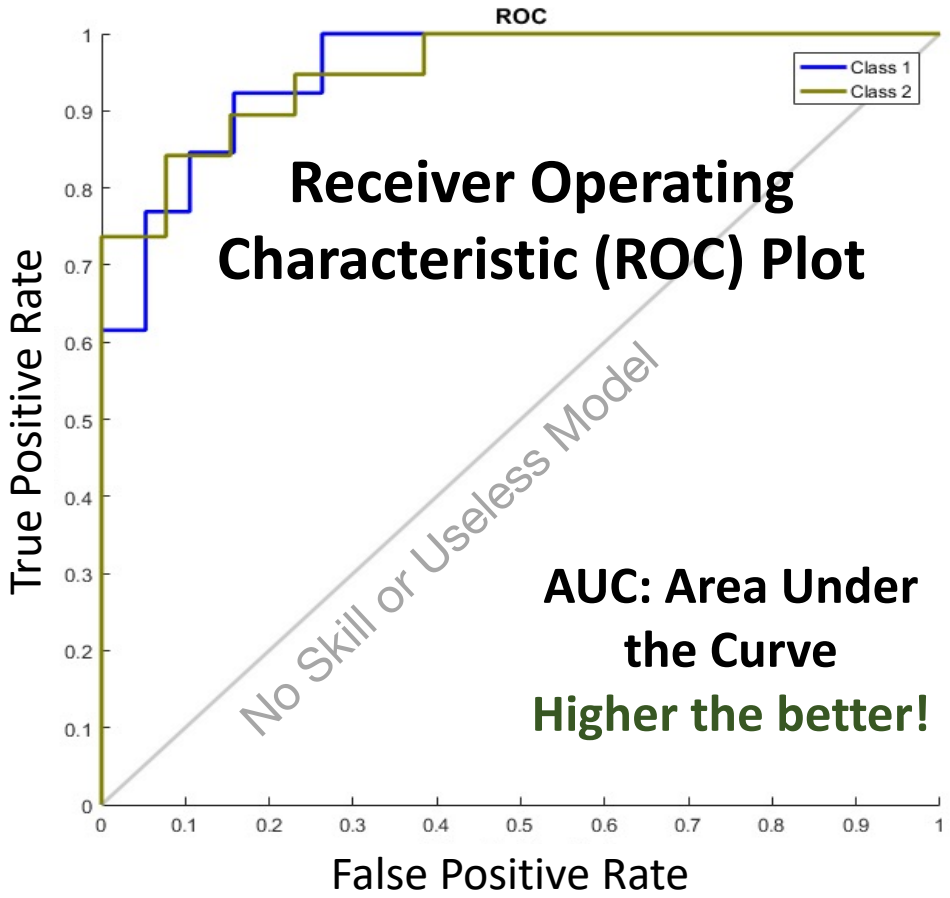
True Condition		Predicted Condition: Predicted +ve	Predicted Condition: Predicted –ve	
	Condition +ve	**True⁺**	**False⁻ (Type-II error)**	**Sensitivity** or **Recall**= $\frac{\sum True^{+}}{\sum Condition^{+}}$
	Condition –ve	**False⁺ (Type-I error)**	**True⁻**	**Specificity** = $\frac{\sum True^{-}}{\sum Condition^{-}}$
	Prevalence $= \frac{\sum Condition^{+}}{\sum Total}$	**Precision** = $\frac{\sum True^{+}}{\sum Predicted^{+}}$	**-ve Predictive Value** $= \frac{\sum True^{-}}{\sum Predicted^{-}}$	**Accuracy** = $\frac{\sum True^{+} + \sum True^{-}}{\sum Total}$





Pattern Recognition (Classifiers): ROC Plot

- Shows how false +ve and true +ve rates change as “threshold” for output classification is varied from 0 to 1
  - Straightforward for binary classification
- Ideally, we are looking for the curve to bow towards the top left corner
  - A completely random model with no utility will yield an ROC plot that is diagonal at 45%
- One should pick a threshold that best balances the two rates to meet the needs of the application
- For multi-class problems, each class will be compared against all other classes
  - One ROC curve for each class (not effective)



Sample ROC Curve Plot for Binary Classification Problem (Cancer Detection)

(Class 1: cancer patients; class 2: normal patients)

`Matlab:-> plotroc(testT,testY)`

Pattern Recognition (Classifiers): Class Imbalance

- **Class Imbalance**: Observations not uniformly distributed across classes
  - Example: Outcome of "strep test" for patients with sore throat condition is mostly "negative" for Streptococcal pharyngitis
- $\boldsymbol{F_1}$ **Score**: Option for binary classification with class imbalance

$$F_1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

- $\boldsymbol{F_1}$ **Score vs. Accuracy**: Weather Prediction Example
  - Suppose only 10% of days are "sunny"
  - A model that (almost) always predicts "not sunny" (say 99.999% of the time, randomly) has an accuracy of 90% whereas it's F_1 score will be 0
    - Precision will be $\approx 0/(0+0)$ NaN and Sensitivity will be $0/(0+10) = 0$

Tactics for Addressing Class Imbalance

- Collect more data
- Try different algorithms
- Change performance metric during model optimization (e.g., F_1 Score)
- Resample your dataset (Link)
  - Over-sampling: Add copies of instances from under-represented class
  - Under-sampling: Delete instances from over-represented class
- Generate synthetic examples
  - Synthetic Minority Over-sampling Technique (SMOTE): Creates synthetic samples from minor class by selecting two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances (Python Module | R DMwR)
- Cost-sensitive learning: Add more cost for minority classes
- Try different perspective: Anomaly detection (e.g., 1-class classification)
- Try getting creative: Redefine the problem or classes!

Pattern Recognition (Classifiers): Cost Sensitivities

- **Setting**: In cases where the cost of false +ves is different from cost of false –ves, "***cost sensitive learning***" will be necessary
  - Example: Cost of a false negative for cancer screening could be a 1,000 times higher than a false positive
- **Vigorous Approach**: Objective for the learning model should explicitly account for differences in costs of different error types
- **Heuristic Approach**: (Binary Classification Example)
  - Model learnt giving equal weight for both types of errors
  - Pick a classifier (threshold) by selecting a point on the ROC that minimizes overall cost

REFERENCE: Elkan (2001) Foundations of Cost-Sensitive Learning | PDF

Time-Series Forecasting

- Partition time-series as consecutive segments for training, validation, and testing (with no overlaps) to demonstrate generalization ability
- MSE plots during training are meaningful as well
- Some popular performance measures:
  - ***MAPE (Mean Absolute Percent Error):***
    - MAPE is scale sensitive and should not be used with low-scale data
      - MAPE will often take on extreme values for small changes
    - MAPE is undefined when Actual value is zero (enters denominator)
  - ***MAD (Mean Absolute Deviation):***
    - Calculated as average of unsigned errors

$$\left(\frac{1}{n}\sum\frac{|Actual - Forecast|}{|Actual|}\right)*100$$

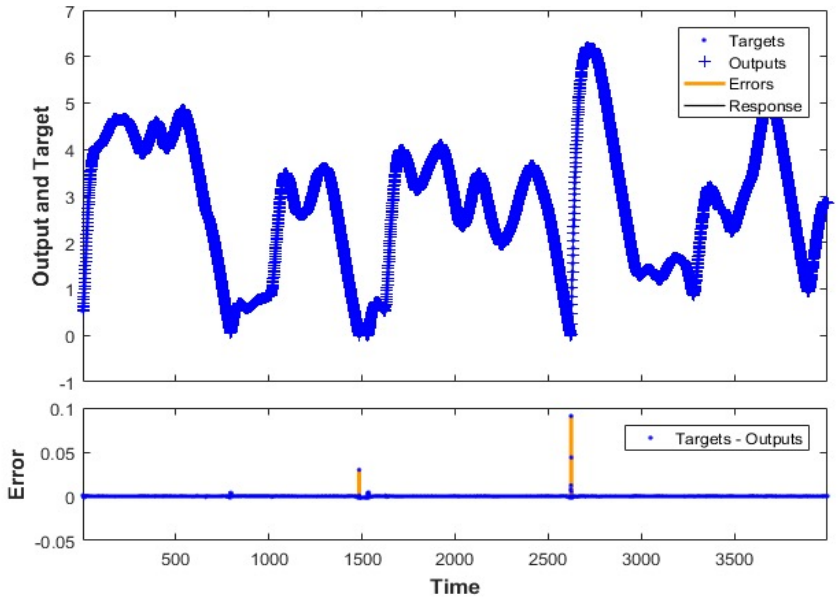
Month	Actual	Forecast	Absolute Percent Error
1	112.3	124.7	11.0%
2	108.4	103.7	4.3%
3	148.9	116.6	21.7%
4	117.4	78.5	33.1%
MAPE			**17.6%**

$$\frac{1}{n}\sum|Actual - Forecast|$$

Month	Actual	Forecast	Absolute Error
1	112.3	124.7	12.4
2	108.4	103.7	4.7
3	148.9	116.6	32.3
4	117.4	78.5	38.9
MAD			**22.08**

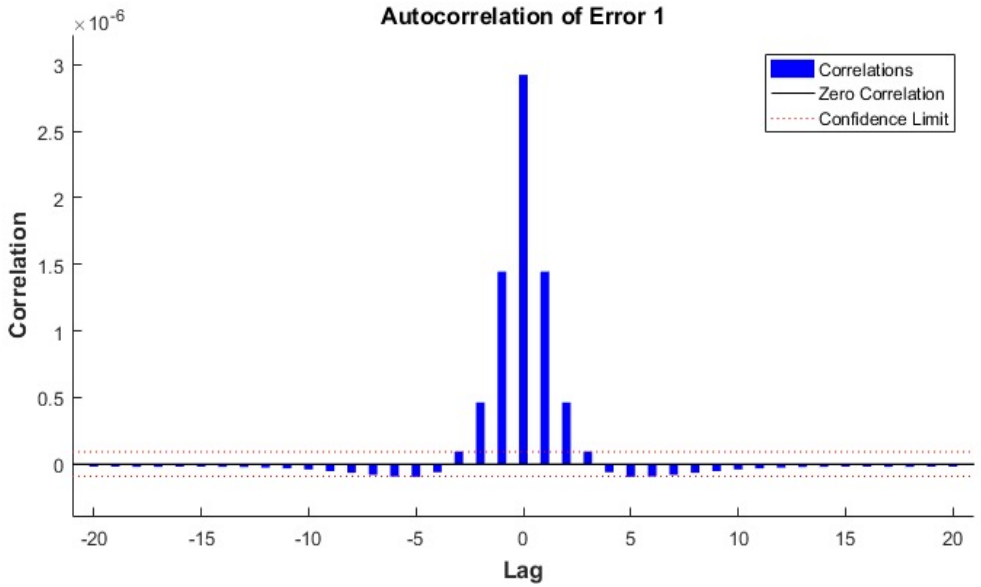
Time-Series Forecasting …

- Plot actual vs predicted values for testing times-series segment along with errors
- Plot correlation of error at time t, $e(t)$ with errors over varying lags, $e(t + lag)$
  - All lines should be short



Sample Forecasting Plot

`Matlab:-> plotresponse(Ts,Y)`



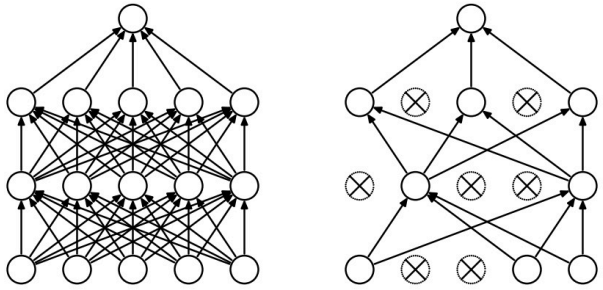
Autocorrelation of Error

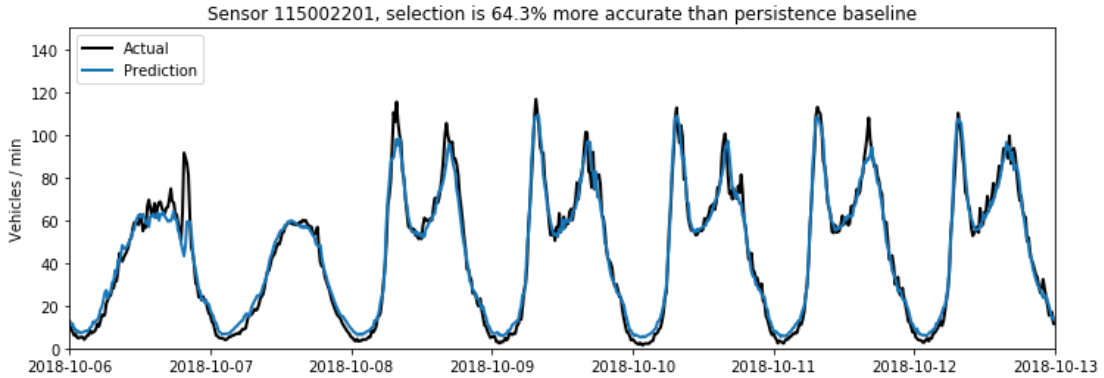
`Matlab:-> E = gsubtract(Ts,Y); ploterrcorr(E)`

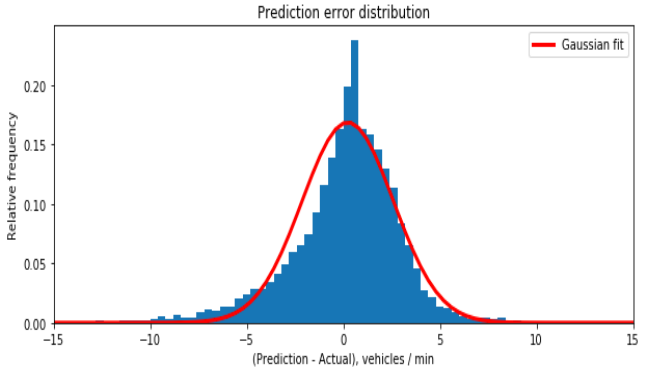
- Model should outperform a naïve model that predicts $\hat{Y}(n+1) = Y(n)$
- Test accuracy of "recursive" predictions if relevant

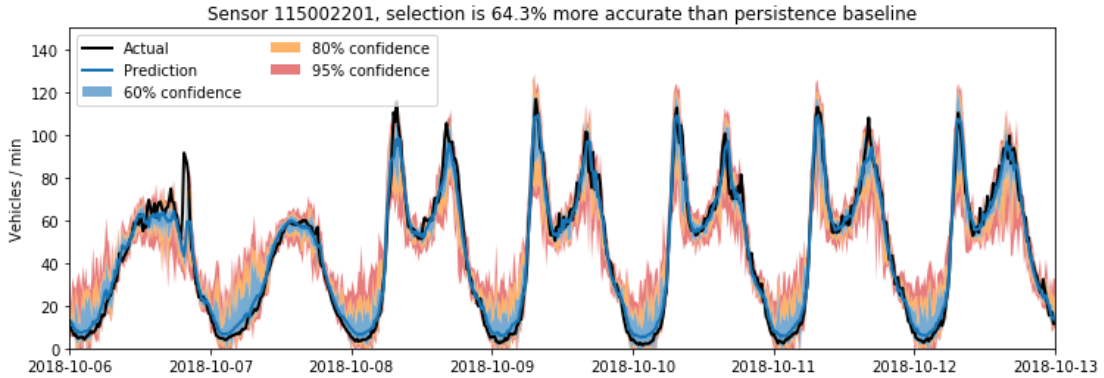
Generating Confidence Intervals using “Dropout” Feature

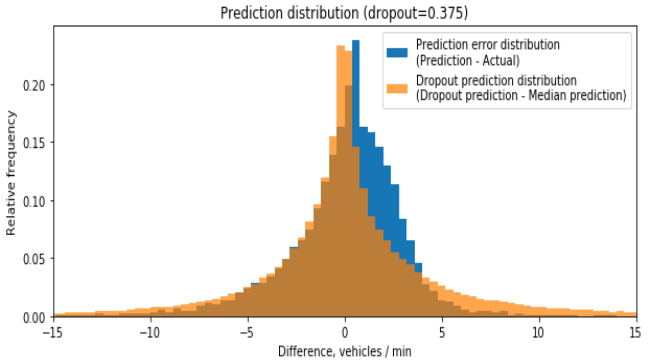
- To get deterministic ANN models to return confidence intervals, exploit “dropout” feature during inference
  - Dropout “turns off” a randomly chosen fraction of model neurons during training to prevent overfitting

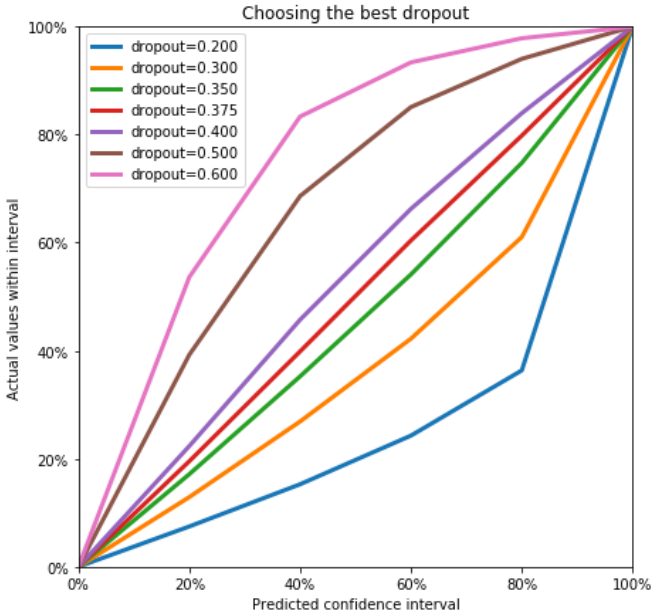












Source: Blake (2019) How to generate neural network confidence intervals with Keras | Link