Rainfall-induced landslide prediction using machine learning: preliminary results and challenges

Saturay[1,2,*], Omadlao[1], Tuguinay[1]

[1]Philippine Science High School - CAR Campus
[2]Engineering Geology Laboratory, NIGS, UP Diliman

*rmsaturay@carc.pshs.edu.ph

Attribution

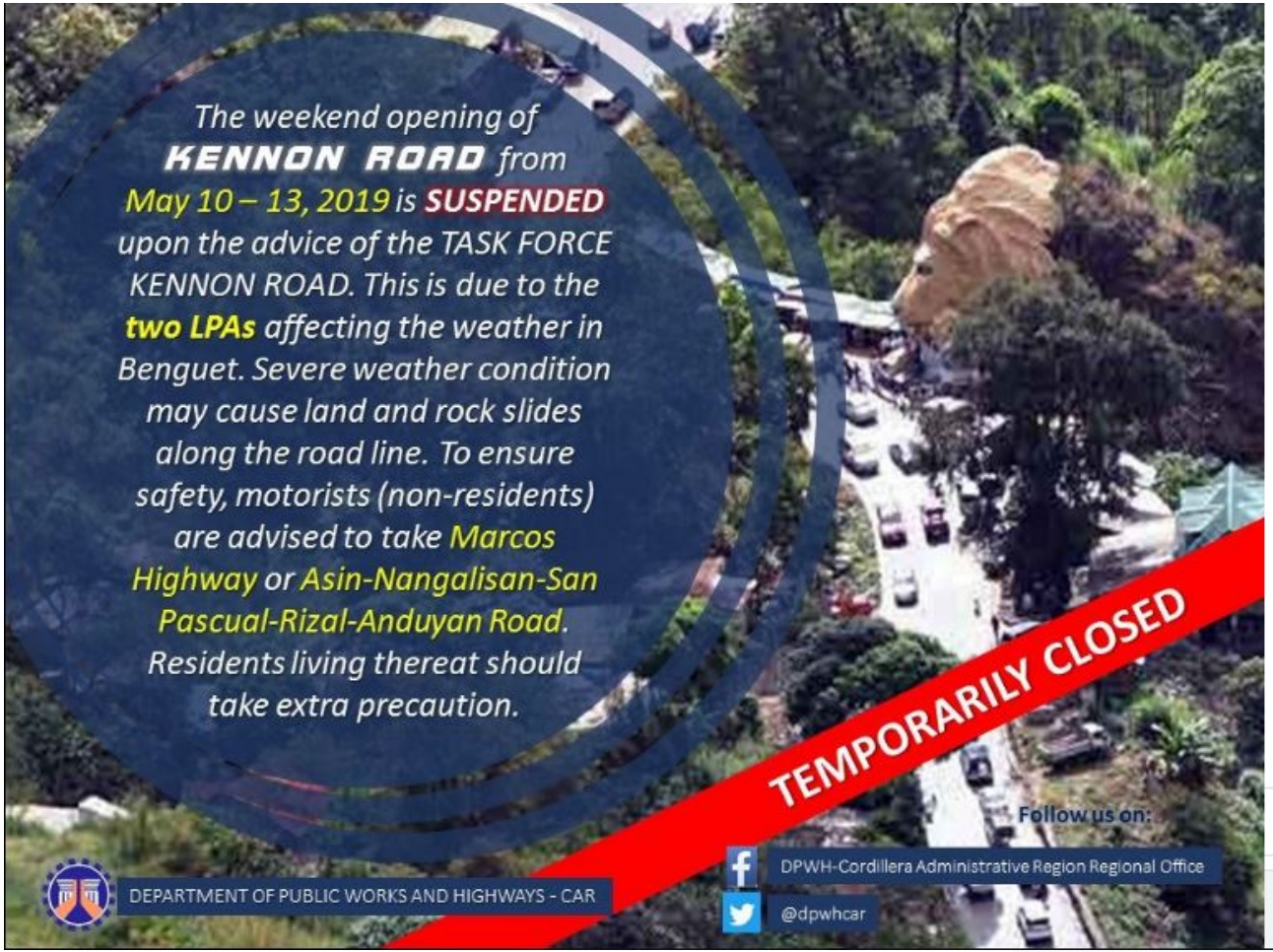
Based on a work at https://doi.org/10.31219/osf.io/csx6r.

Suggested citation:

Saturay, RMJr., Omadlao, ZRD., & Tuguinay, NMA. (2019). Rainfall-induced landslide prediction using machine learning: preliminary results and challenges [presentation slides]. Geocon 2019: Geoscience for a resilient and sustainable Philippines. December 4-5, 2019. Manila, Philippines.

Problem context:

Landslides are major risks along highways in Benguet.



DPWH-Cordillera Administrative Region Regional Office

Page Liked · May 8 ·

TRAVEL ADVISORY
Kennon Road
May 8, 2019

The weekend opening of KENNON ROAD from May 10–13, 2019 is suspended upon the advice of the TASK FORCE KENNON ROAD. This is due to the two LPAs affecting the weather in Benguet. Severe weather condition may cause land and rock slides along the road line. To ensure safety, motorists (non-residents) are advised to take Marcos Highway or Asin-Nangalisan-San Pascual-Rizal-Anduyan Road. Residents living thereat should take extra precaution.

107 35 Comments 214 Shares

Like Comment Share

Most Relevant

Write a comment...

https://web.facebook.com/DPWHCordilleraAdministrativeRegion/

Problem:

Landslide prediction system is needed in managing and mitigating risks along highways.

Possible solutions:

1) Spatial (susceptibility/hazard maps) ✔
2) Temporal (empirical-statistical methods; site-specific instrumentation and monitoring)

Possible solutions:

1) Spatial (susceptibility/hazard maps) ✔️
2) Temporal (empirical-statistical methods ❓ ; site-specific instrumentation and monitoring ❓ ❓)

Possible solutions:

1) Spatial (susceptibility/hazard maps) ✅
2) Temporal (**empirical-statistical methods** ❓;
   site-specific instrumentation and
   monitoring ❓ ❓)

Research objectives

- Develop a machine learning - based system for predicting landslides based on rainfall
  - Generate predictors based on rainfall
  - Train various machine learning models
  - Evaluate prediction results

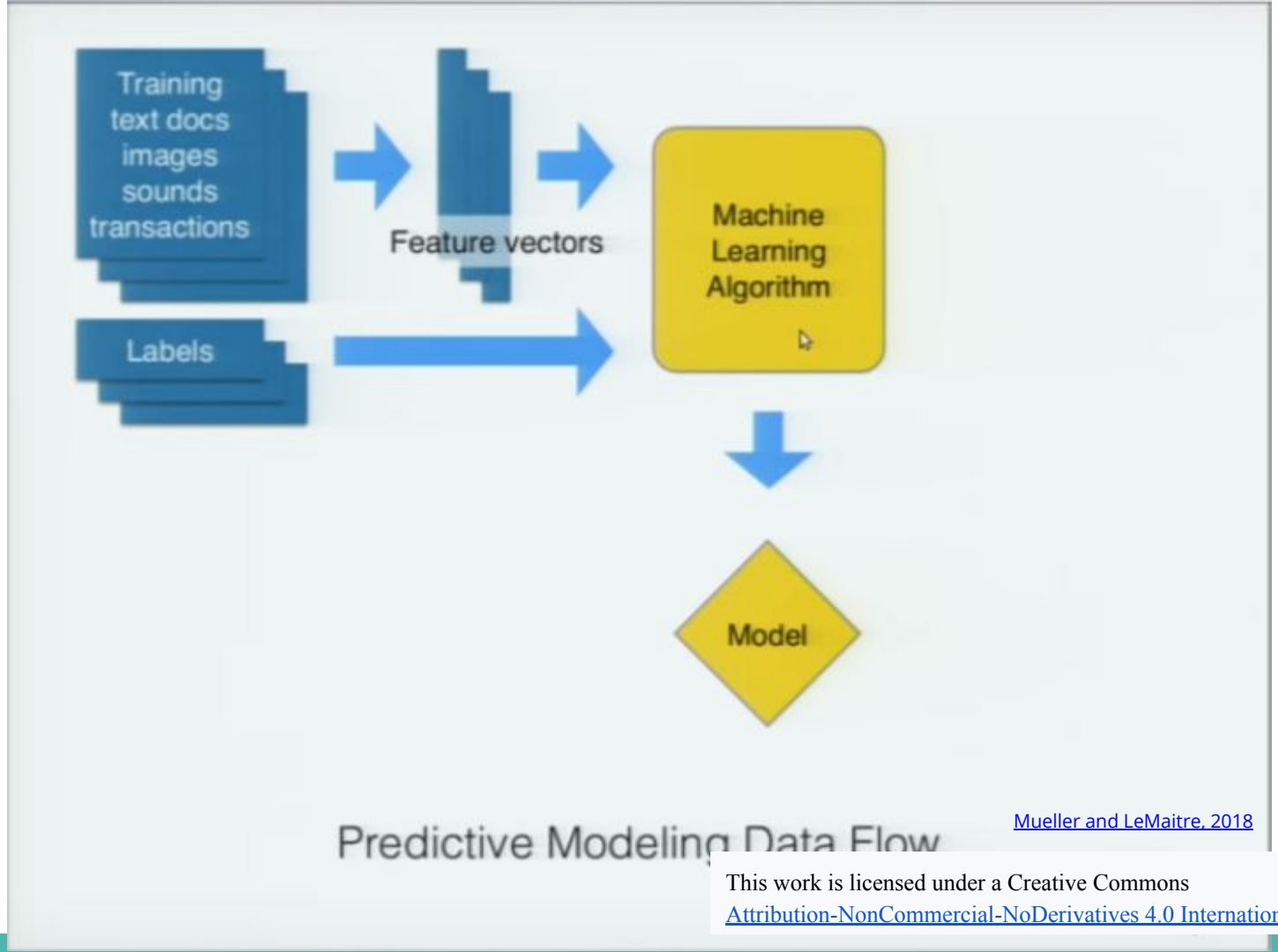
Scope and limitation

- Prediction is for “when”
- Data + ML models
- Default parameters for ML models (blackbox)
- “System” = core function

Presentation objectives

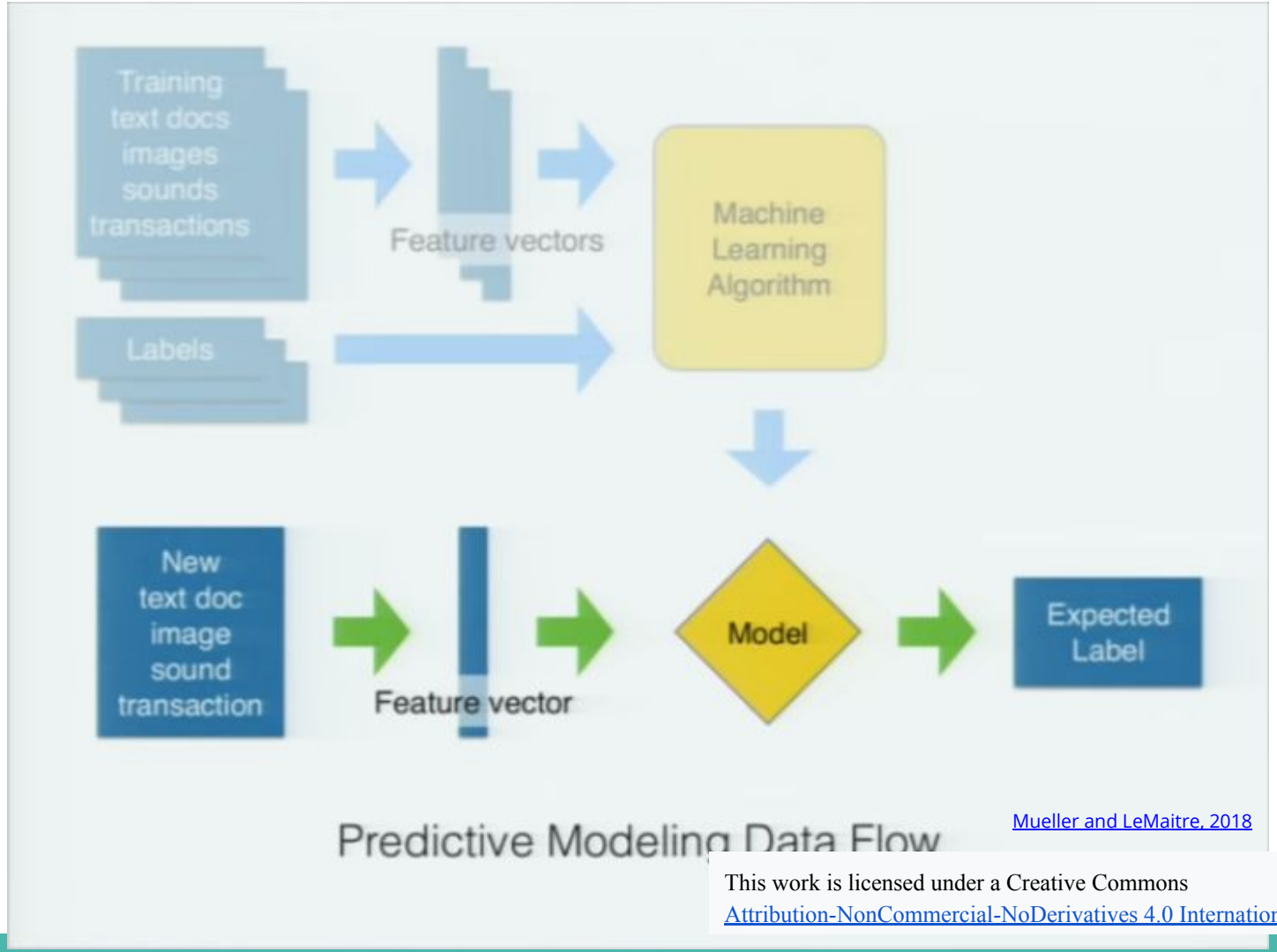
- Discuss (quick) overview of machine learning (ML)
- Demonstrate how ML is applied to the research
- Discuss preliminary results
- Discuss challenges and ways forward

Basic workflow of machine learning (supervised classification)

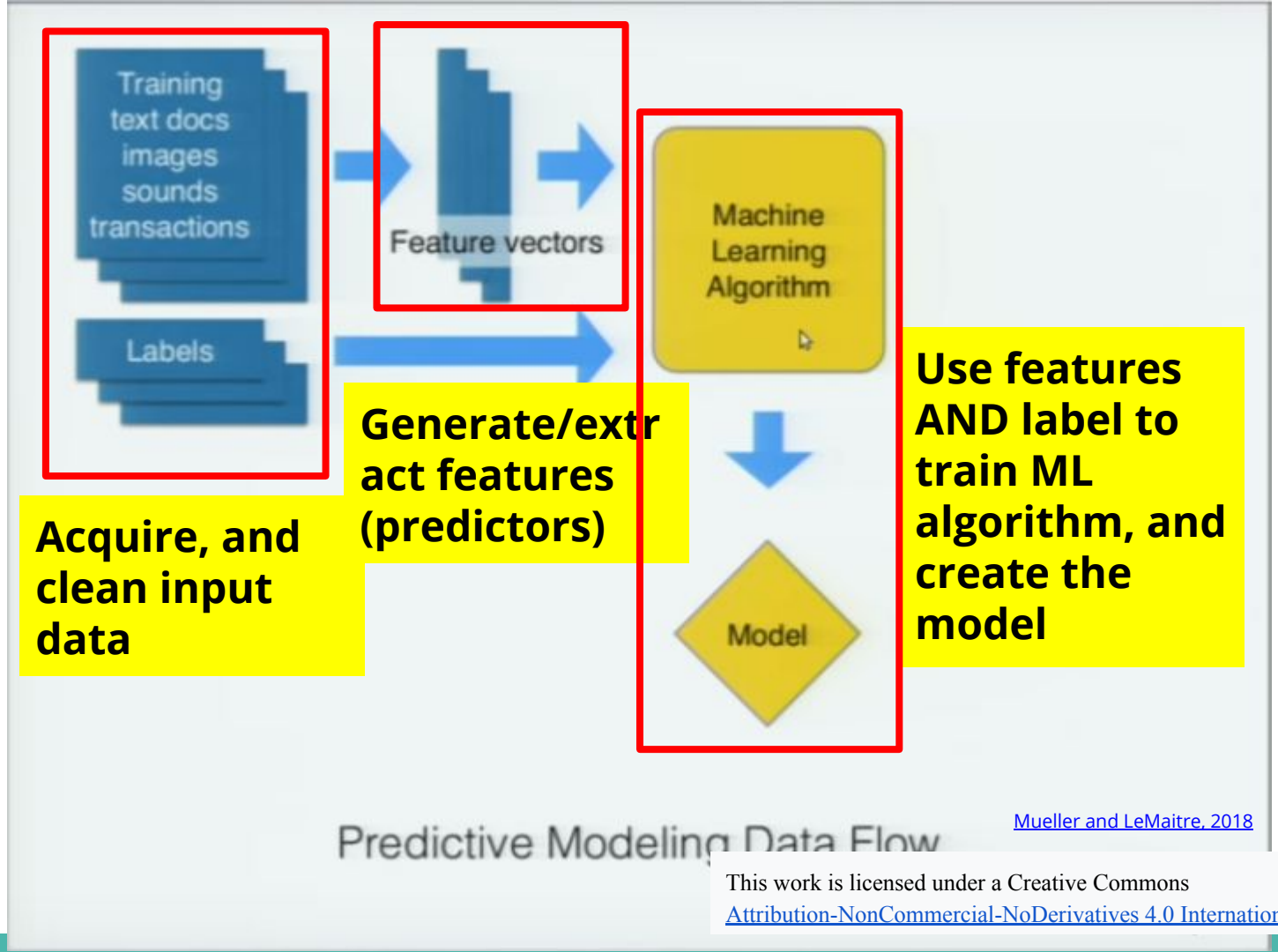


Predictive Modeling Data Flow

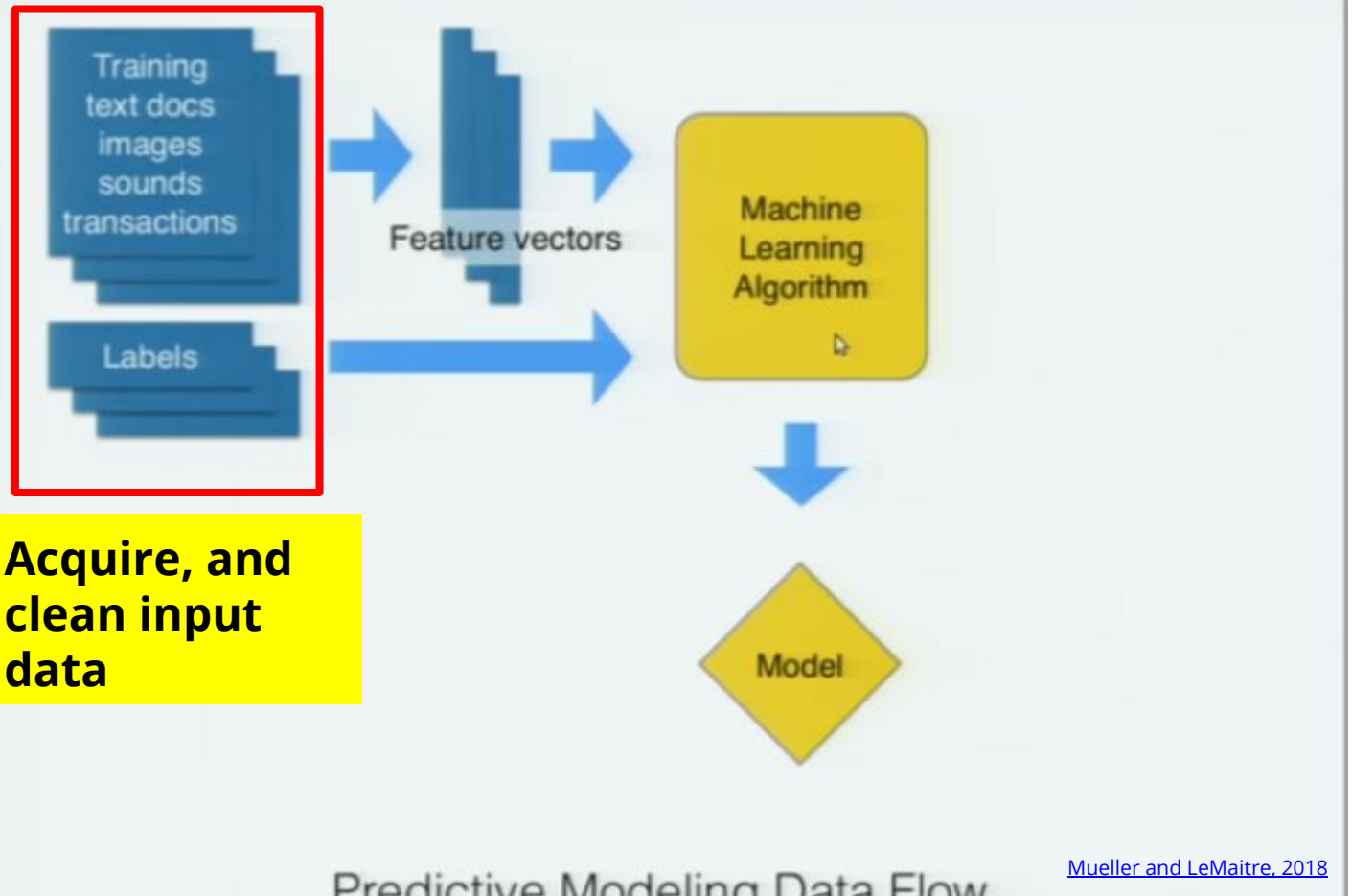
Mueller and LeMaitre, 2018



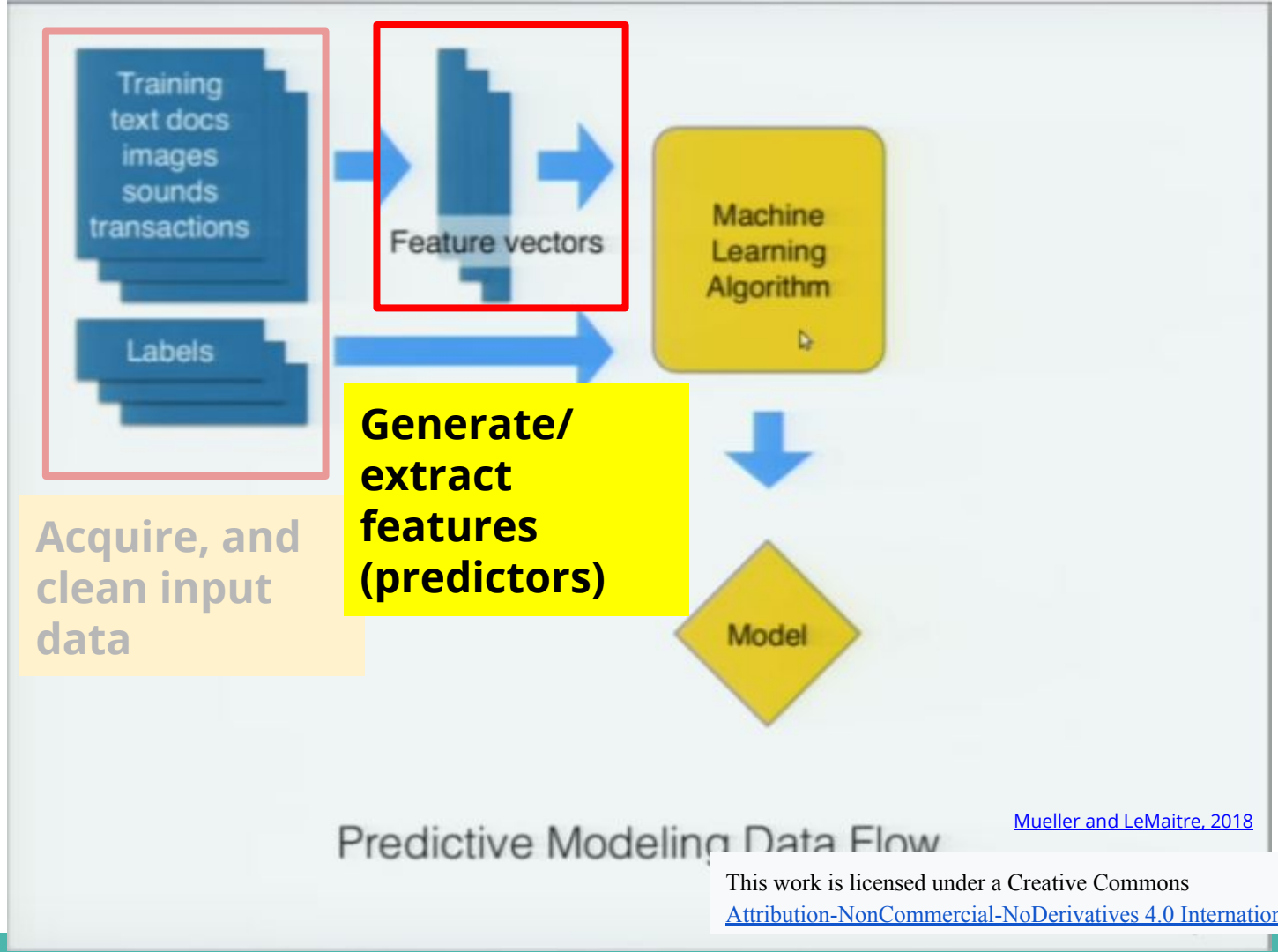
Mueller and LeMaitre, 2018



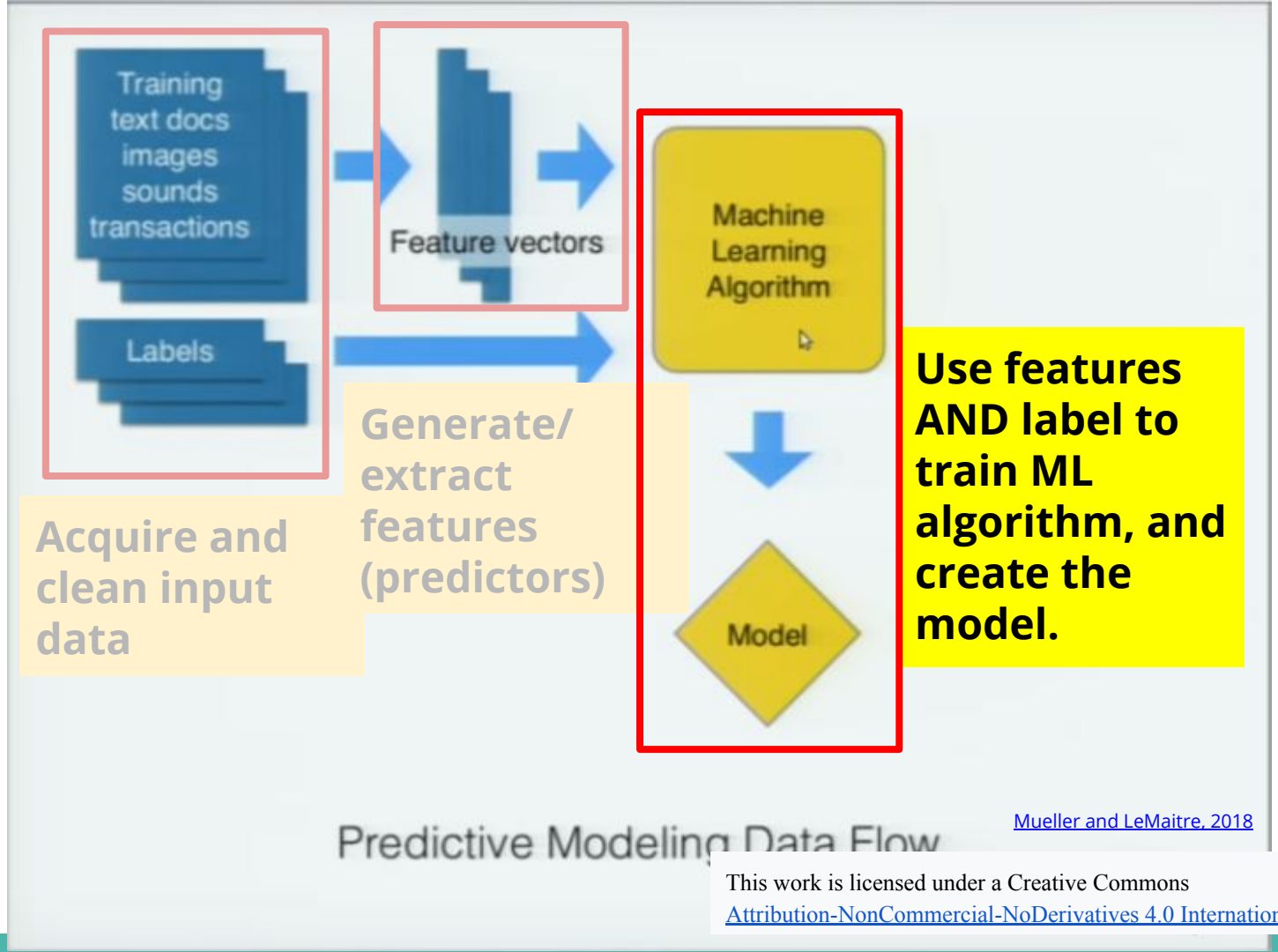
Mueller and LeMaitre, 2018



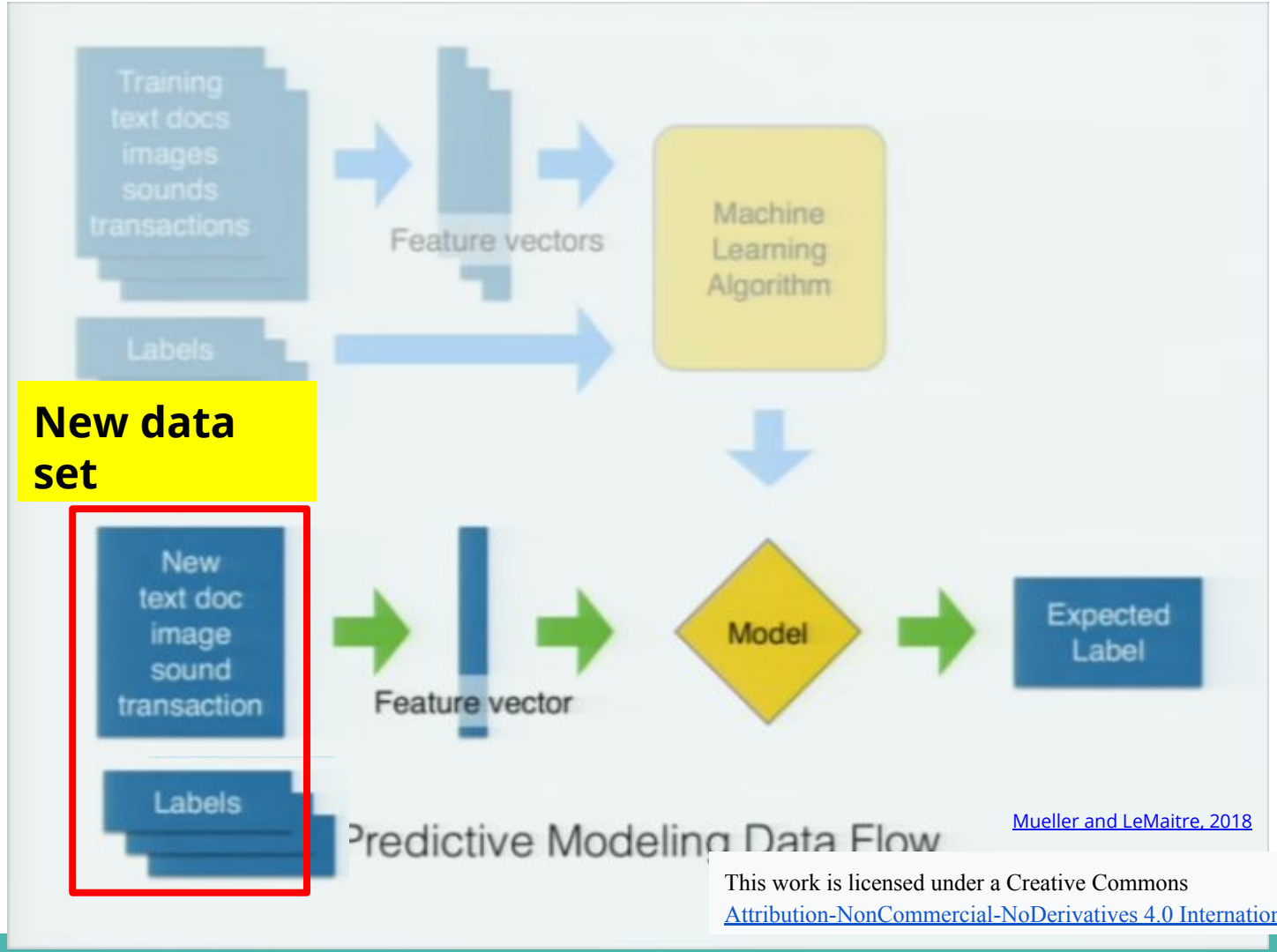
Mueller and LeMaitre, 2018



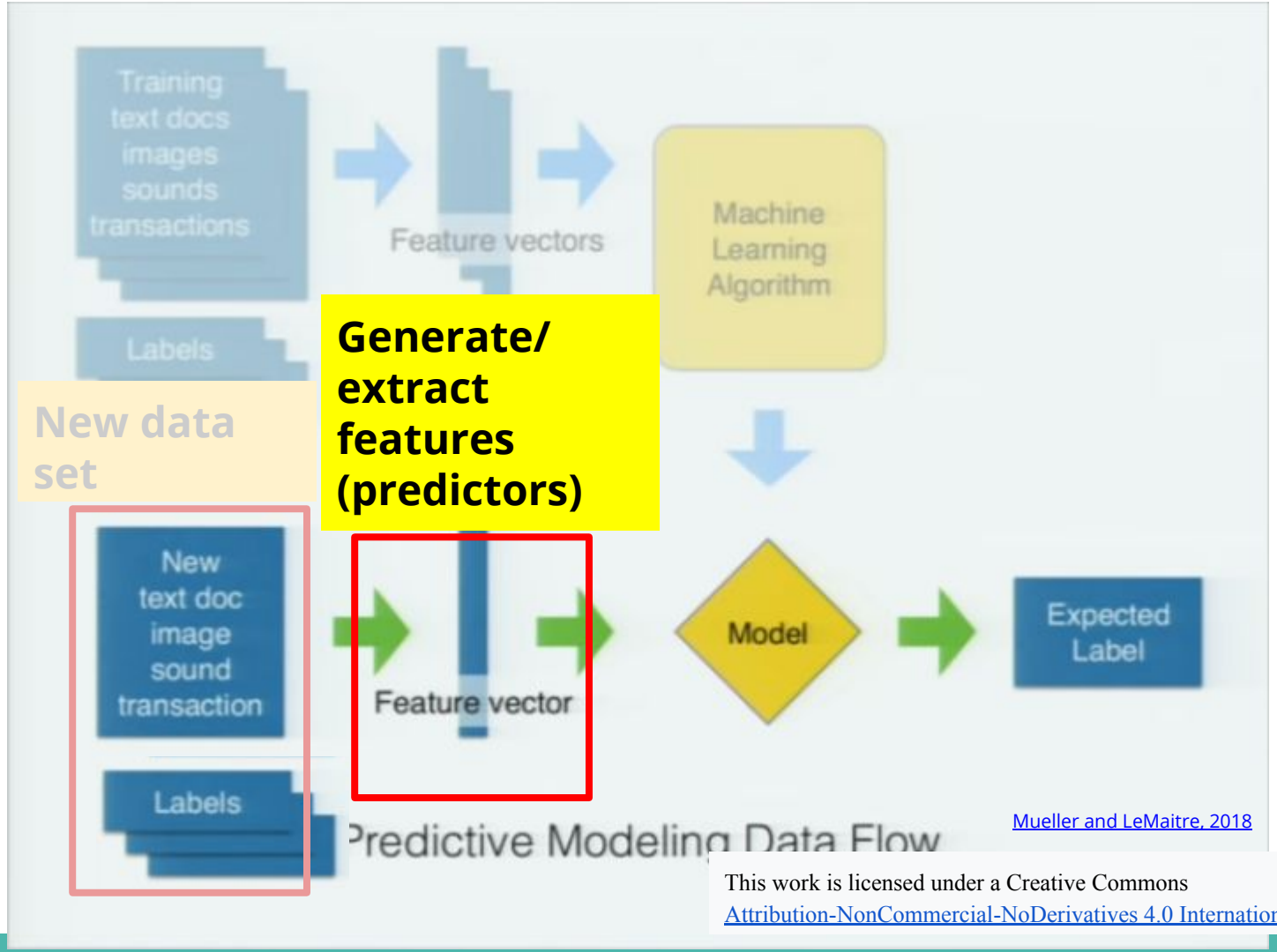
Mueller and LeMaitre, 2018



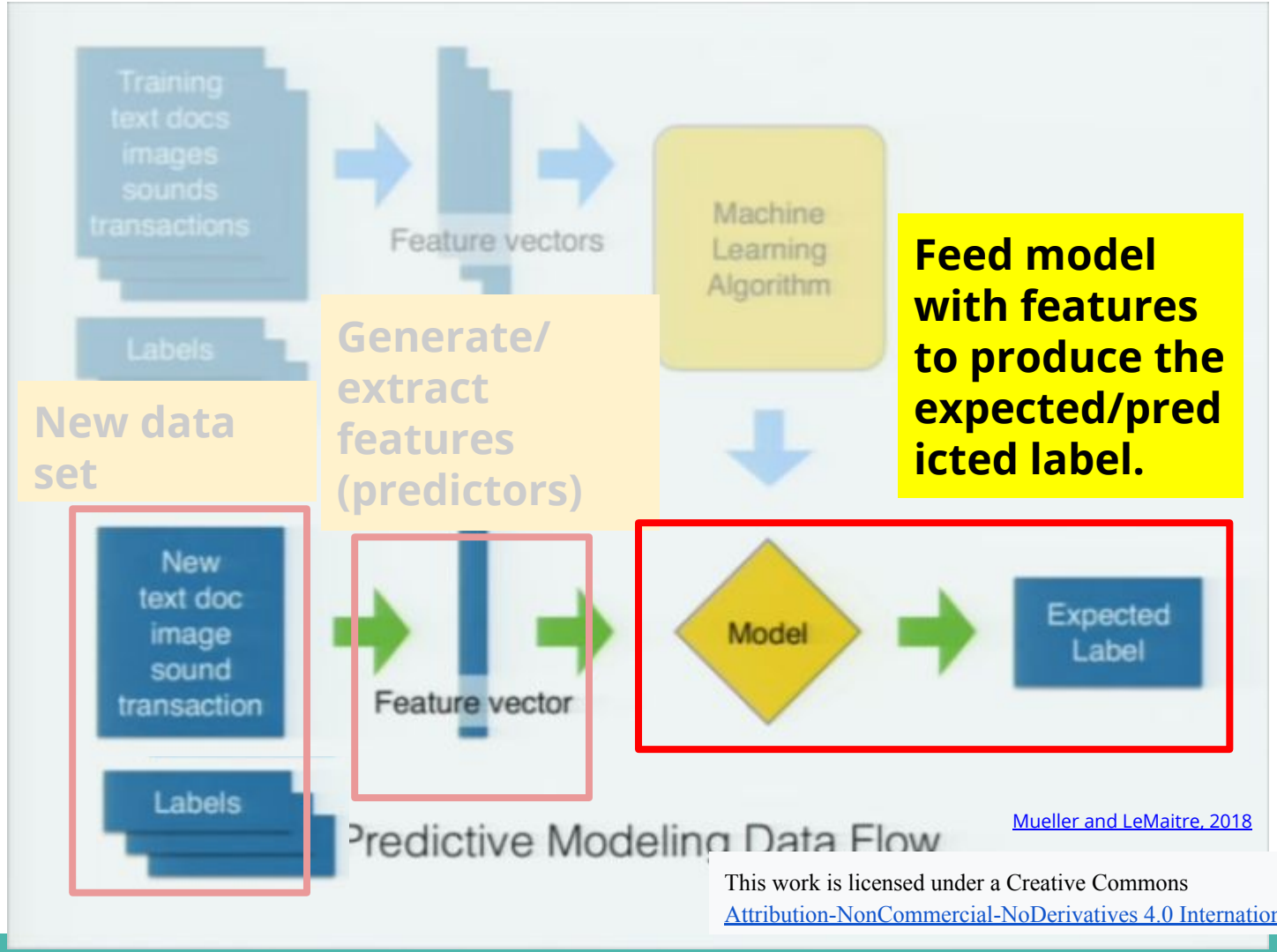
Mueller and LeMaitre, 2018



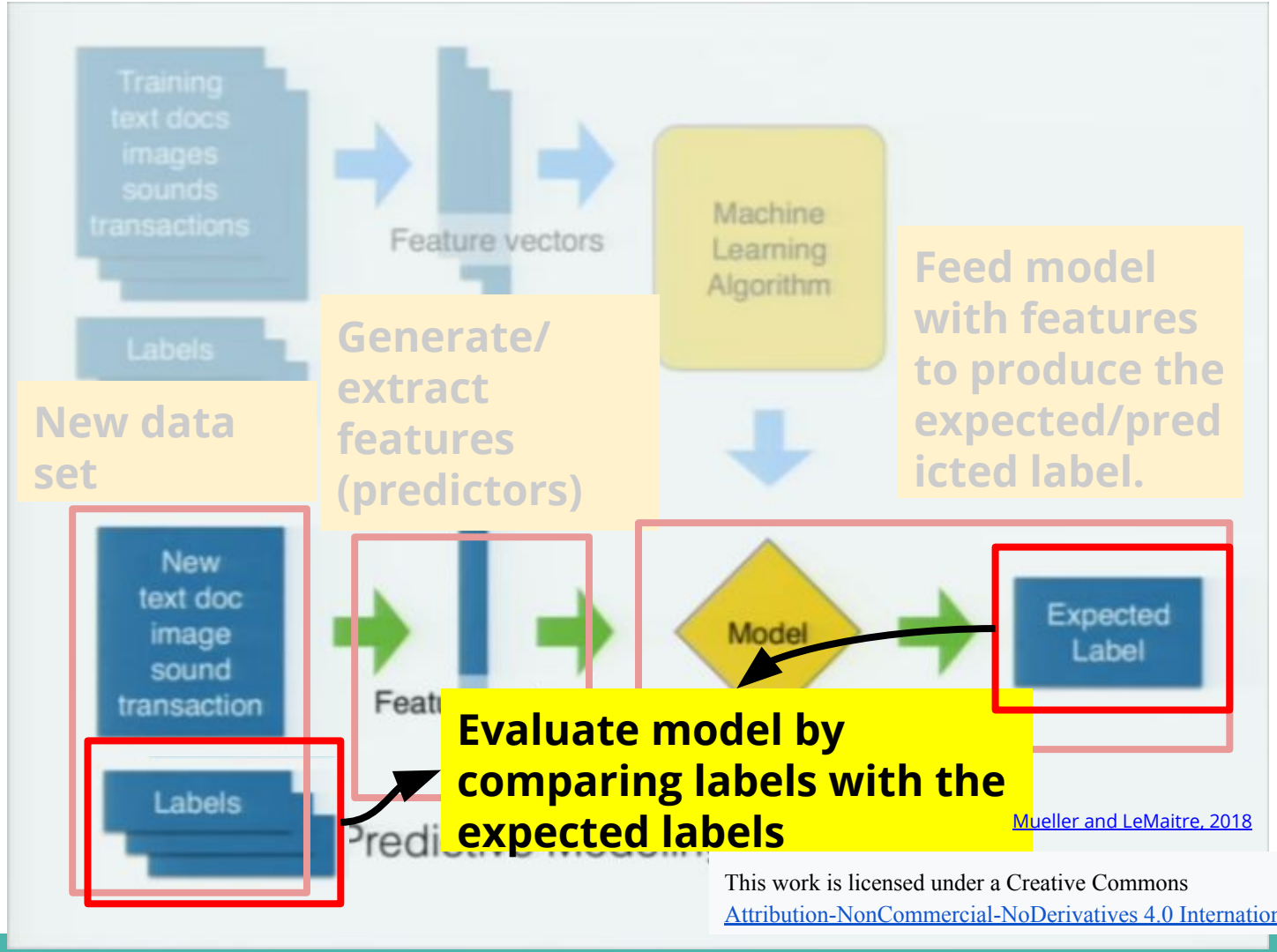
Mueller and LeMaitre, 2018



Mueller and LeMaitre, 2018

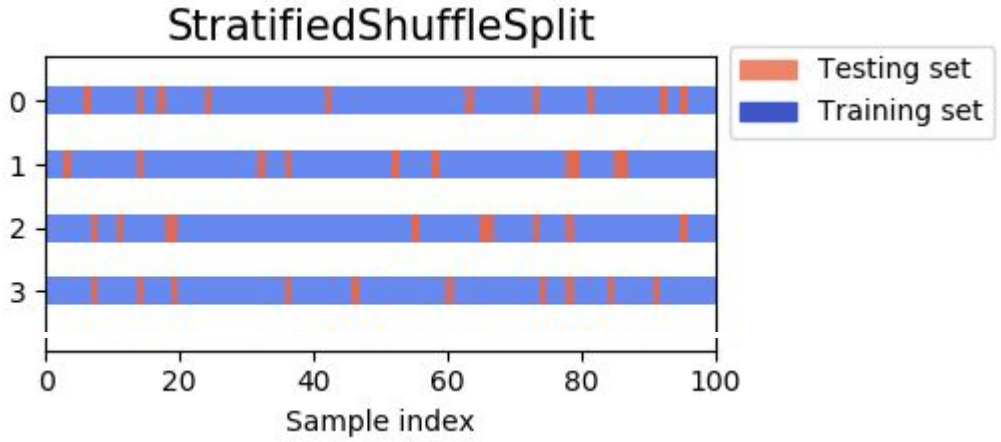


Mueller and LeMaitre, 2018

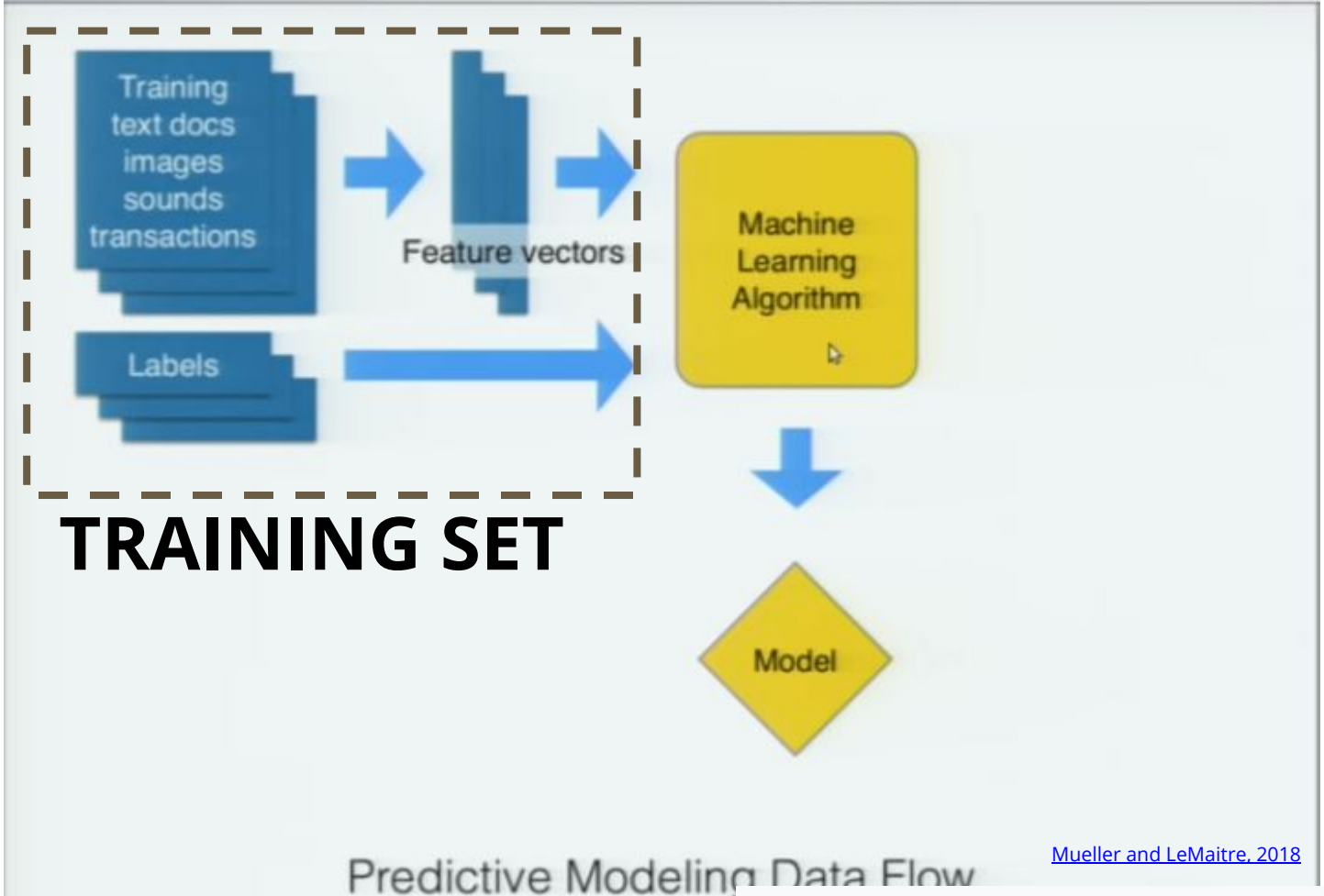


Training vs testing sets

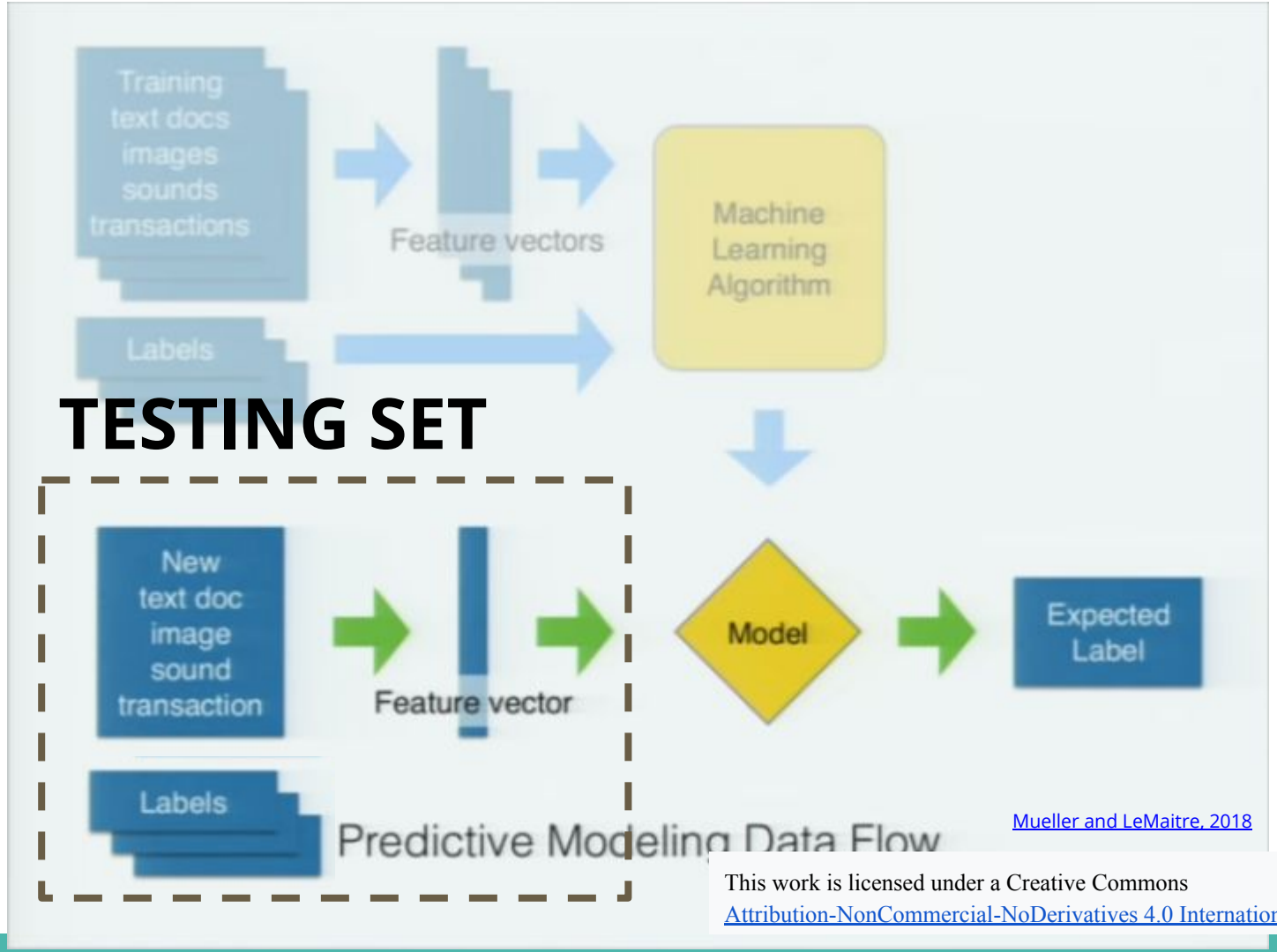
- Common ML practice to split data into training and testing sets.
  - Train the model using the training set.
  - Evaluate model fed with the testing set.



Modified from Buitinck et al., 2013



Mueller and LeMaitre, 2018

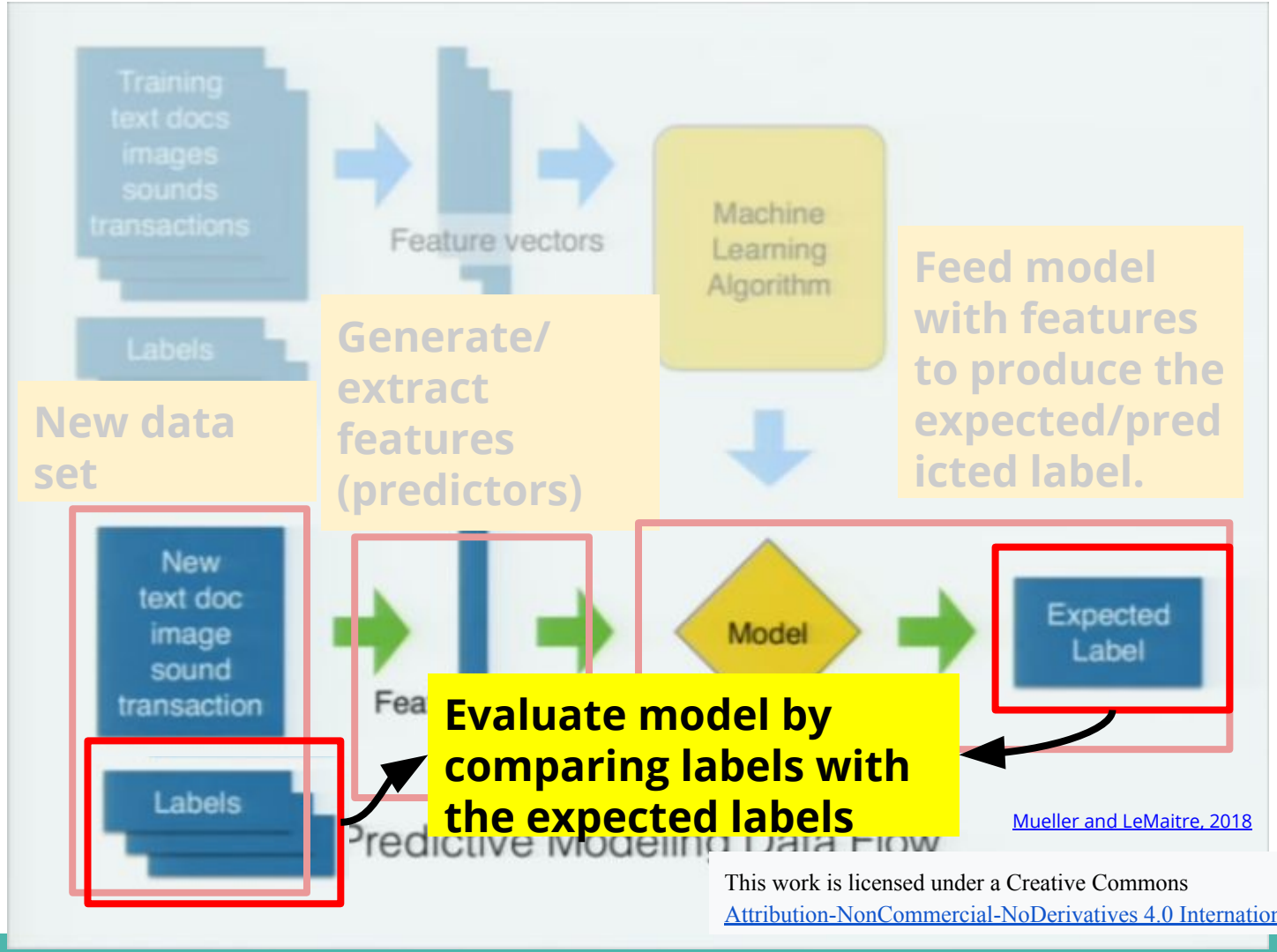


Mueller and LeMaitre, 2018

Model evaluation

- **Confusion matrix**
  - True positive rate
    - TPR=TP/(TP+FN)
  - False positive rate
    - FPR=FP(FP+TN)

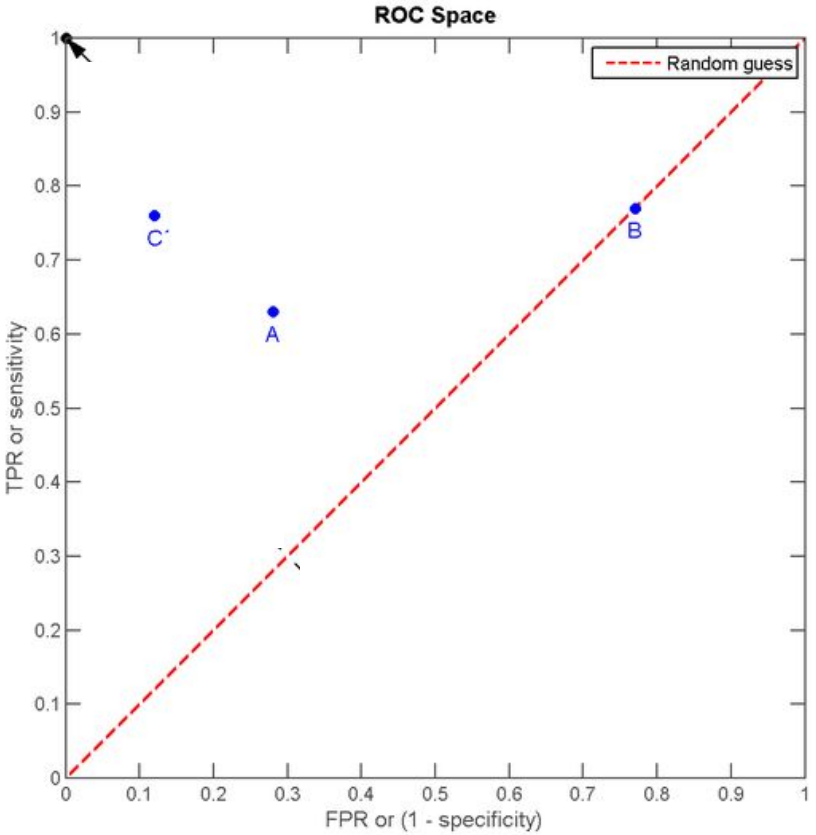
		Expected/Predicted Labels	
		Positive	Negative
Actual Labels	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)



Mueller and LeMaitre, 2018

Model evaluation

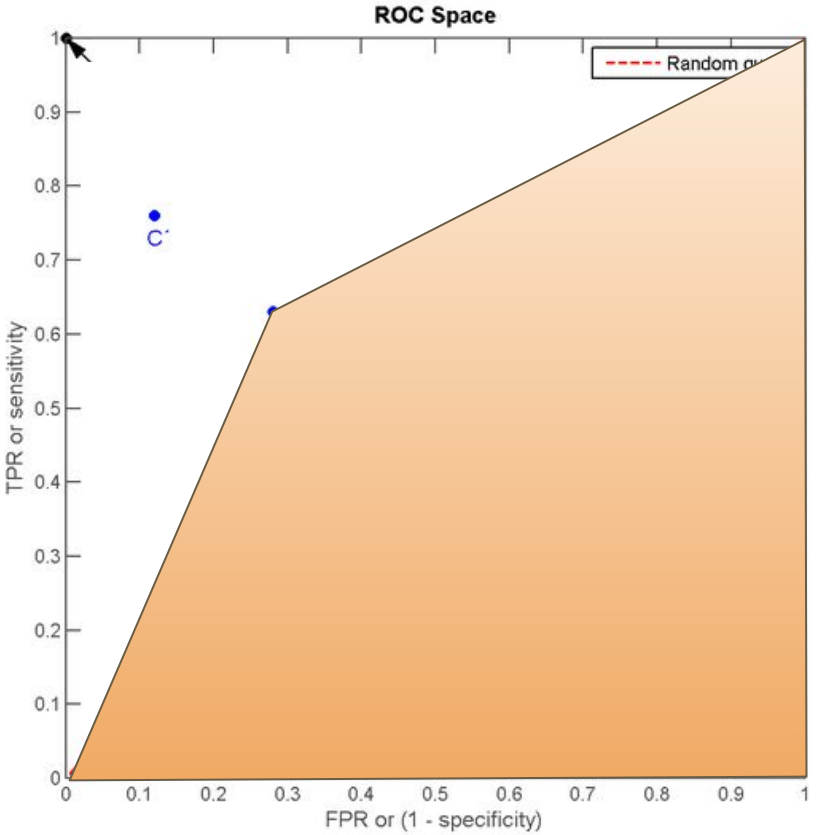
- **Receiver Operating Characteristic (ROC) Curve**
  - **Plot of FPRs vs TPRs**
    - **A, B, C** in graph
  - Area under the ROC curve



Kai walz, 2009. CC-BY-SA-3.0

Model evaluation

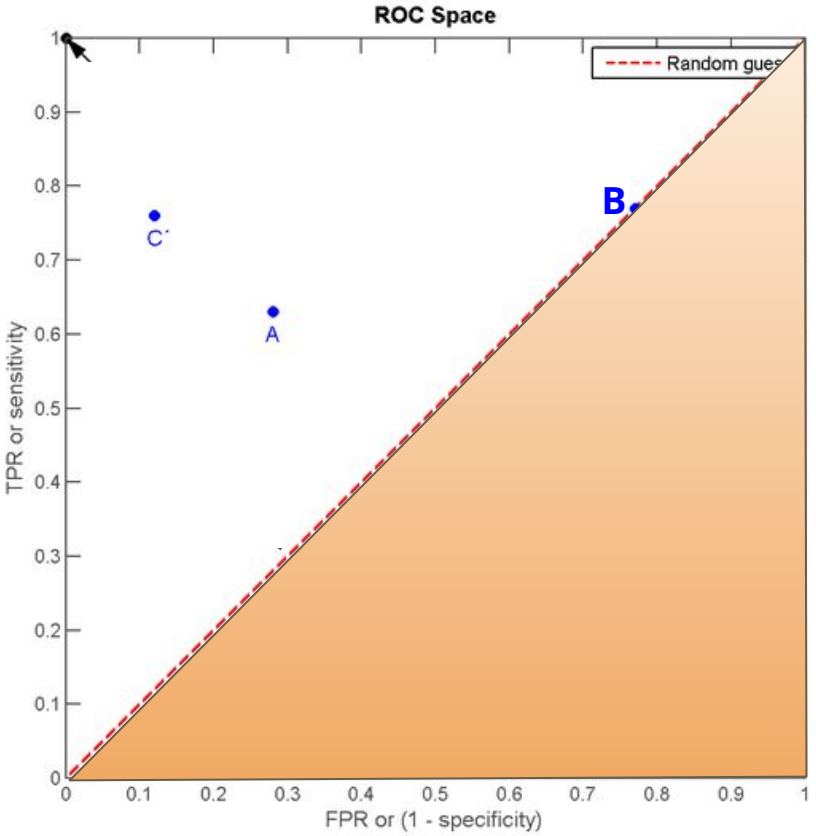
- **Receiver Operating Characteristic (ROC) Curve**
  - Plot of FPR vs TPR
  - **Area under the ROC curve**
    - (0,0)→**$(FPR,TPR)_A$**→(1,1)



Kai walz, 2009. CC-BY-SA-3.0

Model evaluation

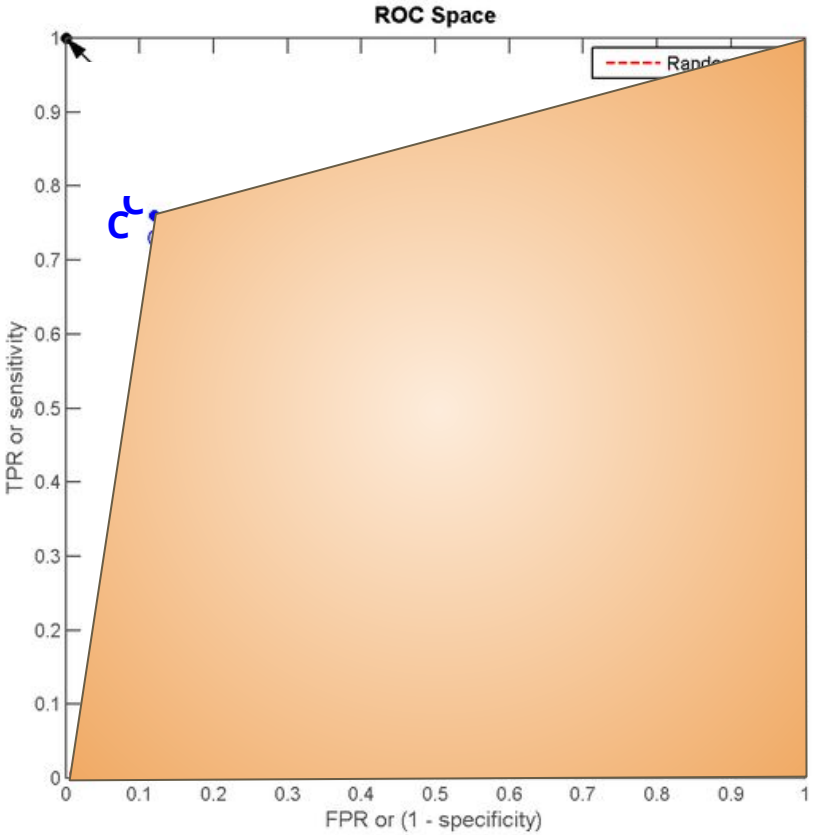
- **Receiver Operating Characteristic (ROC) Curve**
  - Plot of FPR vs TPR
  - **Area under the ROC curve**
    - (0,0)→**$(FPR,TPR)_B$**→(1,1)



Kai walz, 2009. CC-BY-SA-3.0

Model evaluation

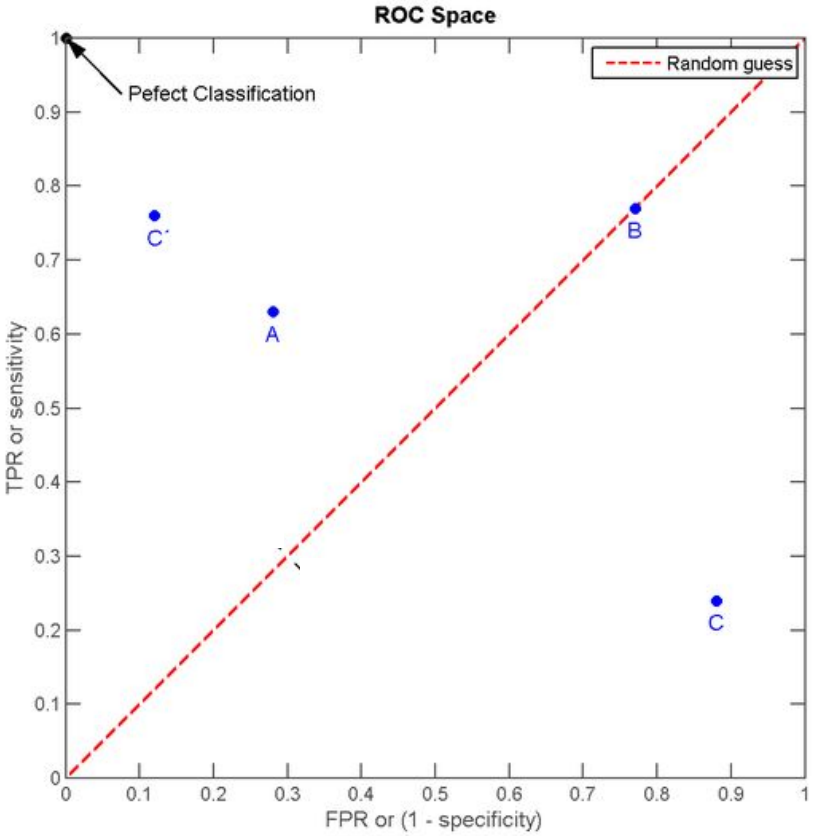
- **Receiver Operating Characteristic (ROC) Curve**
  - Plot of FPR vs TPR
  - **Area under the ROC curve**
    - (0,0)→**(FPR,TPR)$_c$**→(1,1)



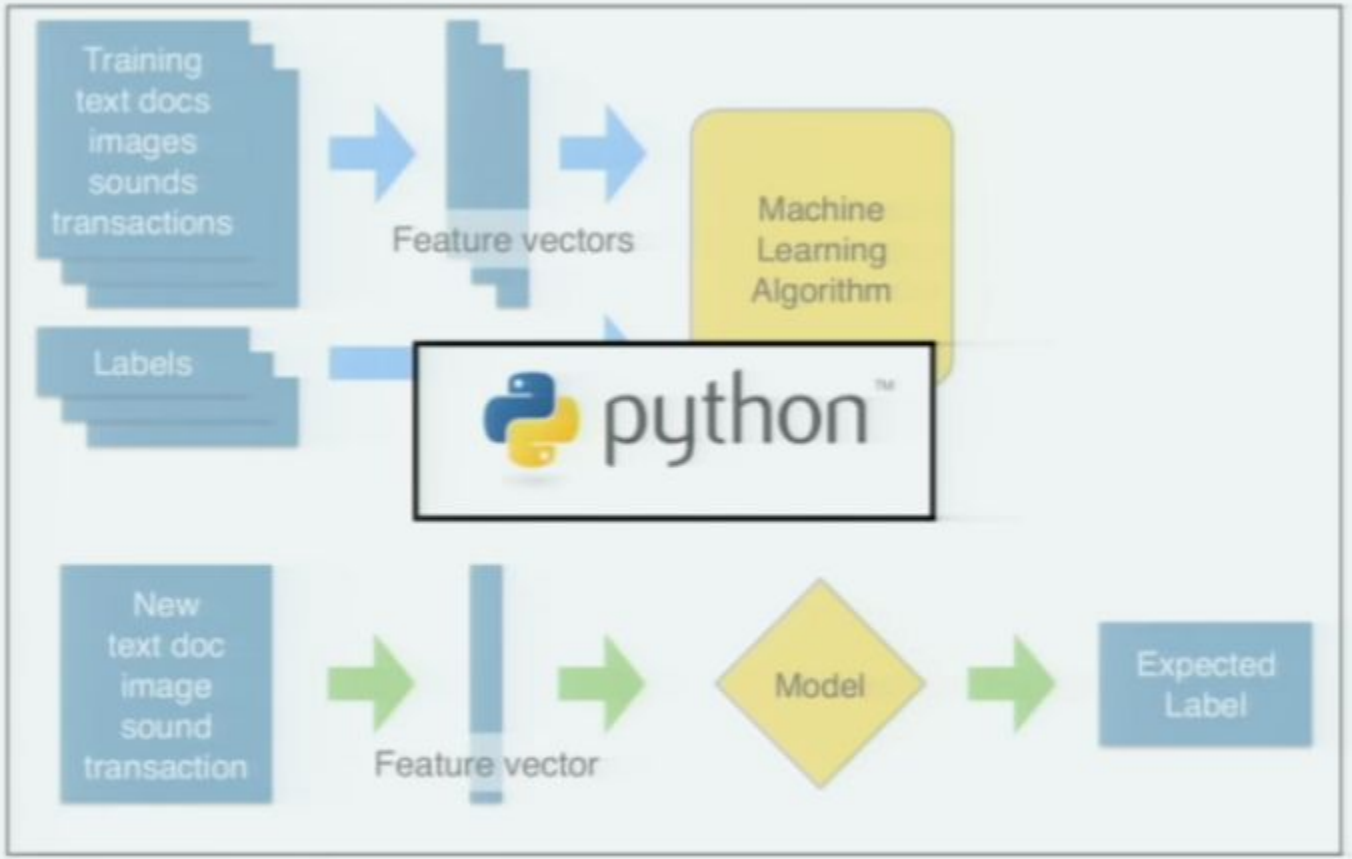
Kai walz, 2009. CC-BY-SA-3.0

Model evaluation

- **Receiver Operating Characteristic (ROC) Curve**
  - Plot of FPR vs TPR
  - Area under the ROC curve
    - Maximum if high TPR, and low FPR

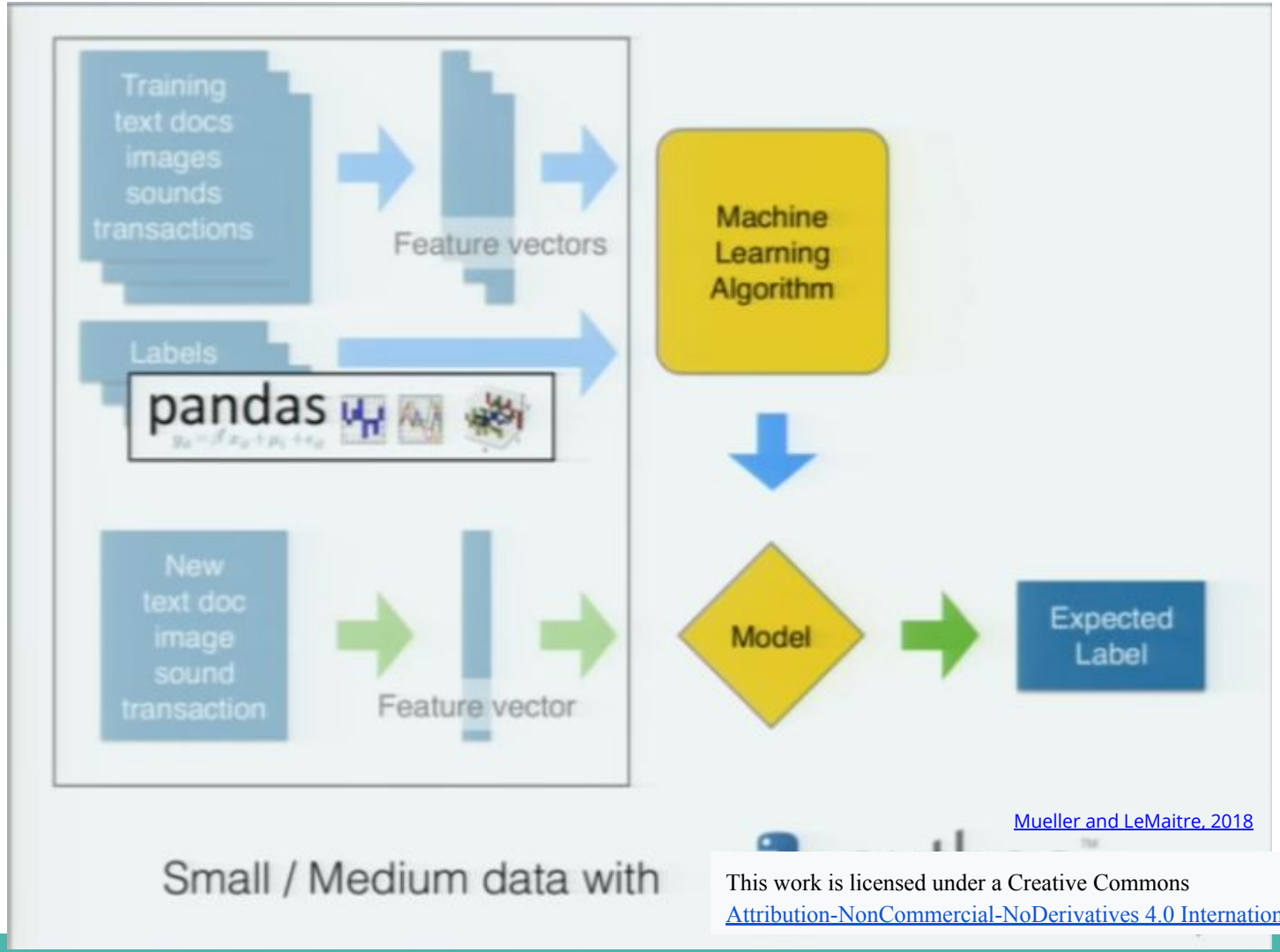


Kai walz, 2009. CC-BY-SA-3.0

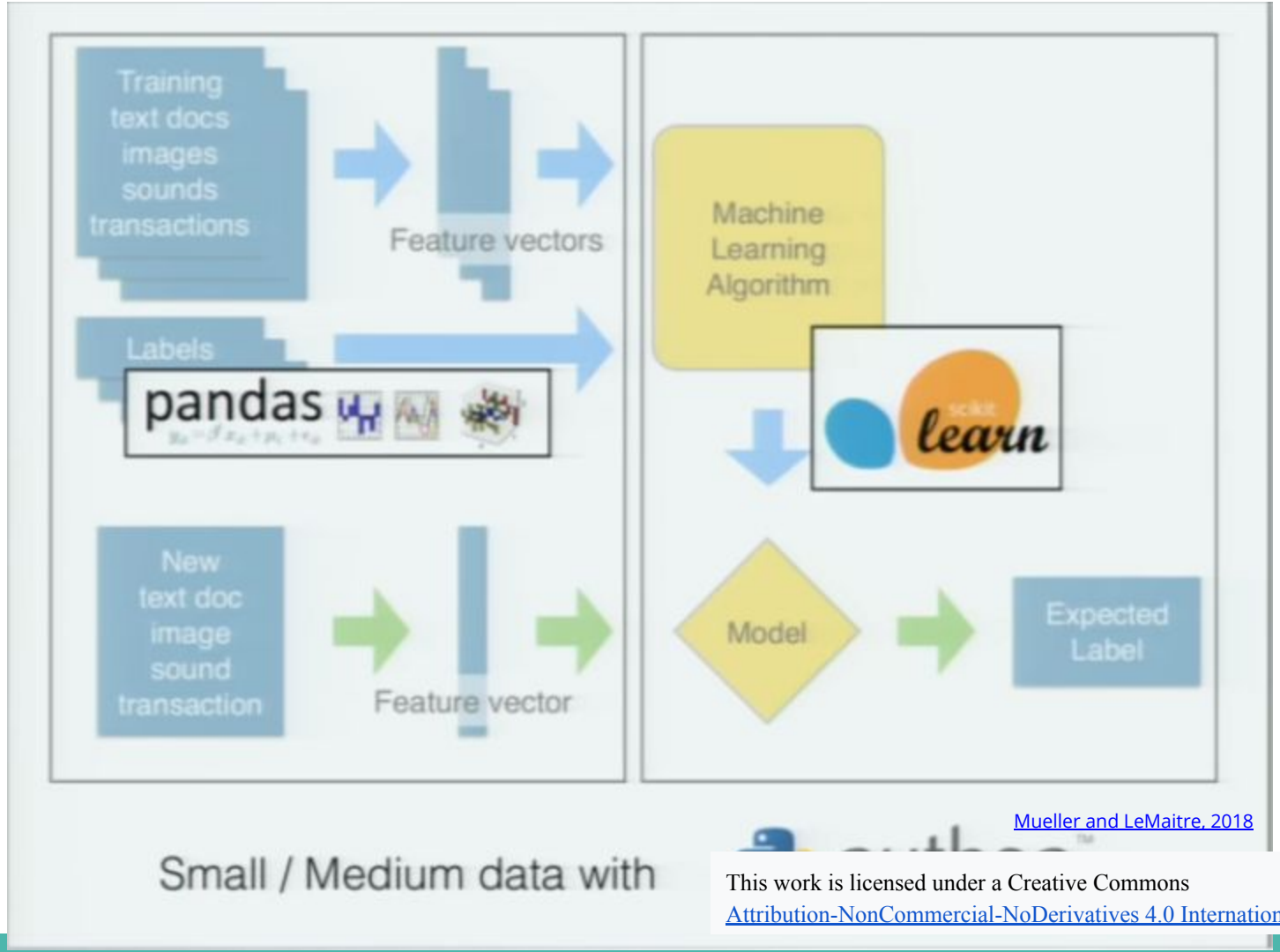


Mueller and LeMaitre, 2018

Small / Mediu



Mueller and LeMaitre, 2018



Mueller and LeMaitre, 2018

Our research...

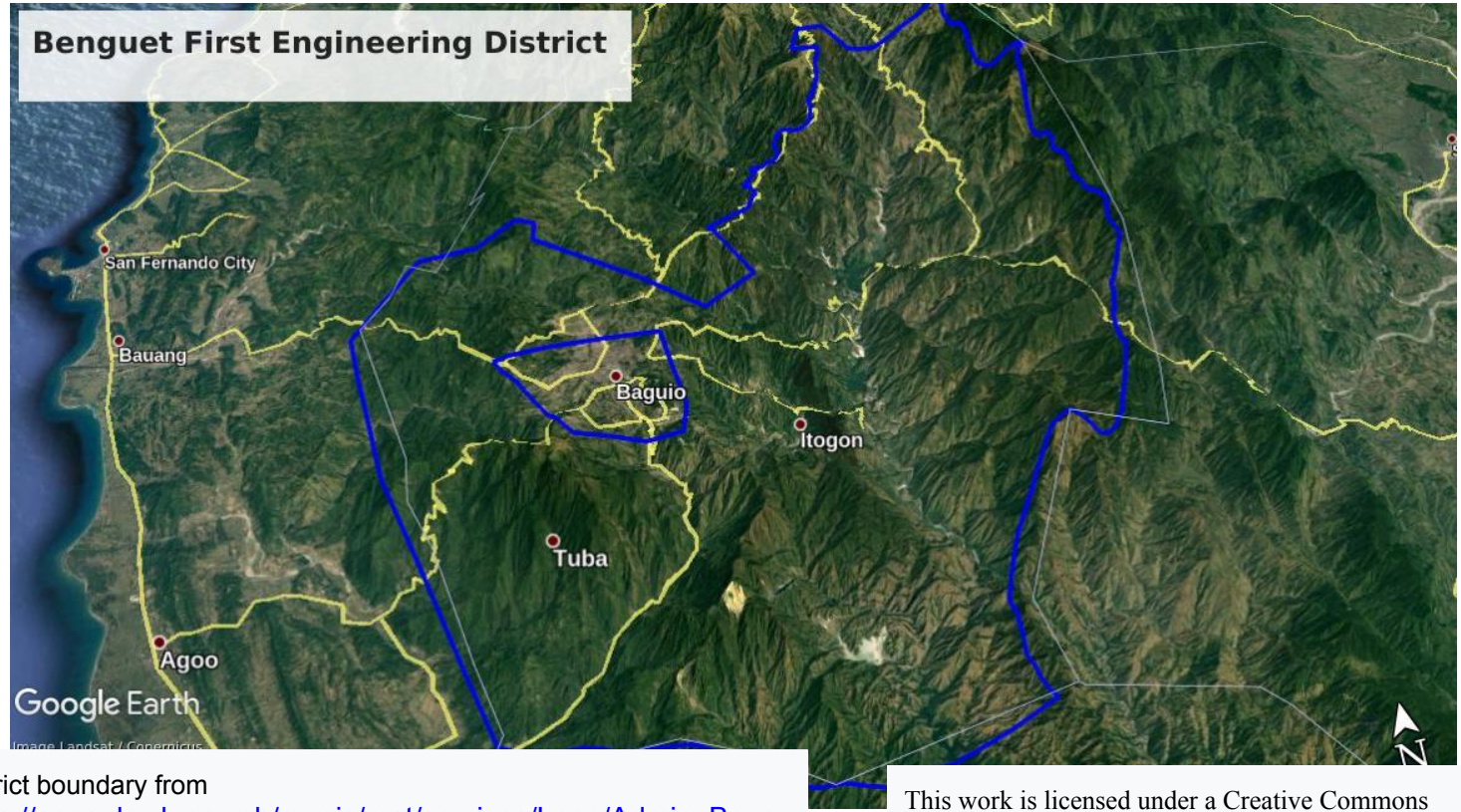
Data acquisition and cleaning → Feature generation → Model training → Model testing → Model evaluation

Years
2014 to 2018

Date	1-day rain (mm)	landslide
…	…	…
4	26	0
5	25	1
6	12	1
7	**10**	**0**

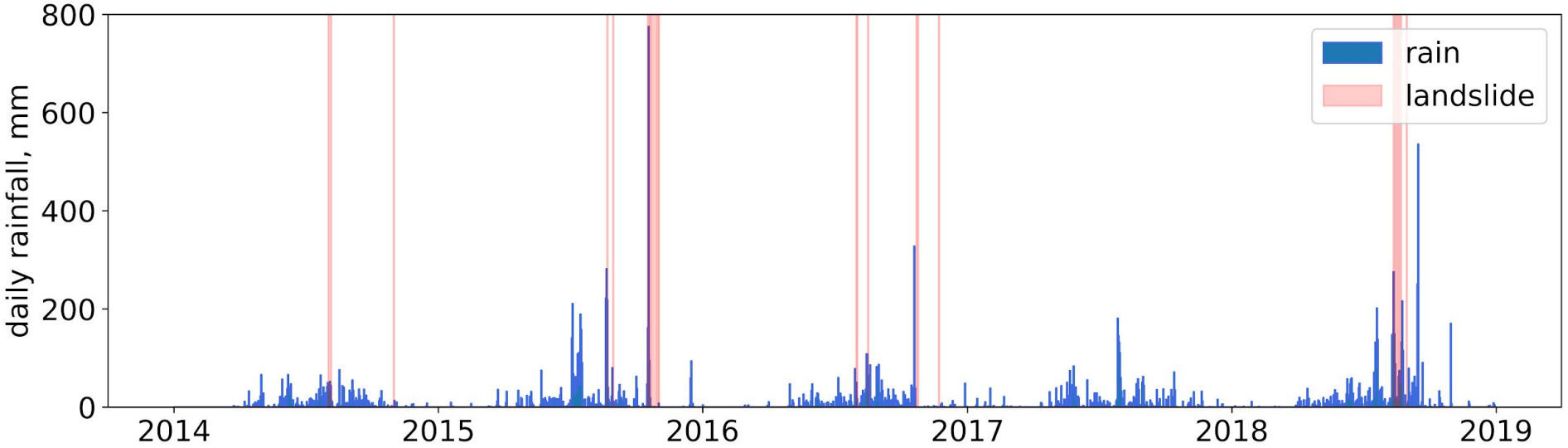
Rainfall data (PAGASA Baguio Synoptic Station);
Landslide data (processed from DPWH-CAR road maintenance records)

Data acquisition and cleaning → Feature generation → Model training → Model testing → Model evaluation



District boundary from https://apps.dpwh.gov.ph/arcgis/rest/services/Leng/Admin_Boundaries_Engineering/MapServer

Data acquisition and cleaning → Feature generation → Model training → Model testing → Model evaluation



Rainfall data (PAGASA Baguio Synoptic Station);
Landslide data (processed from DPWH-CAR road maintenance records)

Cumulative rainfall

Date	1-day rain	2-day rain
1	7	
2	8	15
3	15	23
4	26	41
5	25	51
6	12	37
7	**10**	**22**

SUM

Cumulative rainfall

Date	1-day rain	5-day rain
1	7	
2	8	
3	15	
4	26	
5	25	SUM 81
6	12	86
7	**10**	**88**

Offset rainfall

Date	1-day rain	1-day rain, 1-day offset
1	7	
2	8	7
3	15	8
4	26	15
5	25	26
6	12	25
7	**10**	**12**

OFFSET

Offset rainfall

Date	1-day rain	1-day rain, 3-day offset
1	7	
2	8	
3	15	
4	26	7
5	25	8
6	12	15
7	**10**	**26**

OFFSET

Cumulative + offset rainfall

Date	1-day rain	2-day rain, 3-day offset
1	7	
2	8	
3	15	
4	26	41
5	25	15
6	12	23
7	**10**	**41**

SUM

OFFSET

FEATURES

Date	1-day rain, 0-day offset	5-day rain, 0-day offset	1-day rain, 1-day offset	1-day rain, 3-day offset	2-day rain, 3-day offset	...	Land-slide
7	10	88	12	26	41	...	1

(Cumulative + offset) rainfall + landslide event/non-event label

LABEL

Data acquisition and cleaning → **Feature generation** → Model training → Model testing → Model evaluation

Create features

With the hypothesis that landslide occurrences are determined by certain rainfall characteristics, we create features (columns) from the rainfall time series in our input data. We explore combinations of cumulative and offset functions to generate the various rainfall characteristics.

In [41]:
```
# defining rolling window for cumulative function
cmltv=[1,2,3,5,10,15]
# defining lags for offset function
offset=[0,1,2,3,5]#] #lag
# combining rolling windows and lags
COp=list(itertools.product(cmltv,offset))
#creating dataframe of cumulative-offset rainfalls
COrain=pd.DataFrame(index=rainls.index)
for C,O in COp:
    COrain['c'+str(C)+'_o'+str(O)]=rainls["rain1"].rolling(C).sum().shift(O)
# adding labels to samples (rows)
COrain['ls']=rainls.ls
# removing NaNs as result of cumulative function
COrain=COrain.dropna(axis=0,how='any')
COrain.columns
```

(Cumulative + offset) rainfall + landslide event/non-event label

Data acquisition and cleaning → **Feature generation** → Model training → Model testing → Model evaluation

Create features

With the hypothesis that landslide occurrences are determined by certain rainfall characteristics, we create features (columns) from the rainfall time series in our input data. We explore combinations of cumulative and offset functions to generate the various rainfall characteristics.

In [41]:
```
# defining rolling window for cumulative function
cmltv=[1,2,3,5,10,15]
# defining lags for offset function
offset=[0,1,2,3,5]#] #lag
# combining rolling windows and lags
COp=list(itertools.product(cmltv,offset))
#creating dataframe of cumulative-offset rainfalls
COrain=pd.DataFrame(index=rainls.index)
for C,O in COp:
    COrain['c'+str(C)+'_o'+str(O)]=rainls["rain1"].rolling(C).sum().shift(O)
# adding labels to samples (rows)
COrain['ls']=rainls.ls
# removing NaNs as result of cumulative function
COrain=COrain.dropna(axis=0,how='any')
COrain.columns
```

Cumulative function

Offset function

(Cumulative + offset) rainfall + landslide event/non-event label

Data acquisition and cleaning → **Feature generation** → Model training → Model testing → Model evaluation

```
Out[41]: Index(['c1_o0', 'c1_o1', 'c1_o2', 'c1_o3', 'c1_o5', 'c2_o0', 'c2_o1', 'c2_o2',
                'c2_o3', 'c2_o5', 'c3_o0', 'c3_o1', 'c3_o2', 'c3_o3', 'c3_o5', 'c5_o0',
                'c5_o1', 'c5_o2', 'c5_o3', 'c5_o5', 'c10_o0', 'c10_o1', 'c10_o2',
                'c10_o3', 'c10_o5', 'c15_o0', 'c15_o1', 'c15_o2', 'c15_o3', 'c15_o5',
                'ls'],
               dtype='object')
```

(Cumulative + offset) rainfall + landslide event/non-event label

define classifiers, and corresponding parameters

(modified from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)

In [3]:
```
classifierNames=pd.DataFrame(["kNN", "Linear SVM", "RBF SVM","DT", "RF",
                              "Neural Net", "AdaBoost","GNB","LR"])

classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1),
    AdaBoostClassifier(),
    GaussianNB(),
    LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial')]
```

Models (CLASSIFIERS) selected for training

Splitting data for training and testing

In [32]:
```
n_splits=200
test_size=0.25
random_state=0

# define train-test-split function
sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=random_state)

TPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
FPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
aROCs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])

# loop through all splits
for s,(train_index, test_index) in zip(TPRs.index,sss.split(COrain.iloc[:,:-1], COrain.iloc[:,-1])):
    print(s)
#     define training and testing sets
    ytrain, ytest = COrain['ls'].iloc[train_index], COrain['ls'].iloc[test_index]
    Xtrain, Xtest = COrain[COrain.columns[:-1]].iloc[train_index], COrain[COrain.columns[:-1]].iloc[test_index]
#     print(len(ytrain),len(ytest))
#     normalize training set features (column-wise)
    Xtrain_norm, Xnorm=normalize(Xtrain,axis=0,return_norm=True)
#     normalize testing set features (column-wise)
    Xtest_norm=Xtest/Xnorm
#     print(s, len(ytest[ytest==1]),sum(ytest[ytest==1].index))
    for c in range(len(classifierNames)):
```

Split data into training and testing sets

Splitting data for training and testing

In [32]:
```
n_splits=200
test_size=0.25
random_state=0

# define train-test-split function
sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=random_state)

TPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
FPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
aROCs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])

# loop through all splits
for s,(train_index, test_index) in zip(TPRs.index,sss.split(COrain.iloc[:,:-1], COrain.iloc[:,-1])):
    print(s)
#     define training and testing sets
    ytrain, ytest = COrain['ls'].iloc[train_index], COrain['ls'].iloc[test_index]
    Xtrain, Xtest = COrain[COrain.columns[:-1]].iloc[train_index], COrain[COrain.columns[:-1]].iloc[test_index]
#     print(len(ytrain),len(ytest))
#     normalize training set features (column-wise)
    Xtrain_norm, Xnorm=normalize(Xtrain,axis=0,return_norm=True)
#     normalize testing set features (column-wise)
    Xtest_norm=Xtest/Xnorm
#     print(s, len(ytest[ytest==1]),sum(ytest[ytest==1].index))
    for c in range(len(classifierNames)):
```

Splitting function

Split data into training and testing sets

Splitting data for training and testing

In [32]:
```
n_splits=200
test_size=0.25
random_state=0

# define train-test-split function
sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=random_state)

TPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
FPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
aROCs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])

# loop through all splits
for s,(train_index, test_index) in zip(TPRs.index,sss.split(COrain.iloc[:,:-1], COrain.iloc[:,-1])):
    print(s)
#     define training and testing sets
    ytrain, ytest = COrain['ls'].iloc[train_index], COrain['ls'].iloc[test_index]
    Xtrain, Xtest = COrain[COrain.columns[:-1]].iloc[train_index], COrain[COrain.columns[:-1]].iloc[test_index]
#     print(len(ytrain),len(ytest))
#     normalize training set features (column-wise)
    Xtrain_norm, Xnorm=normalize(Xtrain,axis=0,return_norm=True)
#     normalize testing set features (column-wise)
    Xtest_norm=Xtest/Xnorm
#     print(s, len(ytest[ytest==1]),sum(ytest[ytest==1].index))
    for c in range(len(classifierNames)):
```

Splitting function

Training set (y=label, X=features)

Split data into training and testing sets

Splitting data for training and testing

In [32]:
```
n_splits=200
test_size=0.25
random_state=0

# define train-test-split function
sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=random_state)

TPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
FPRs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])
aROCs=pd.DataFrame(index=range(n_splits),columns=classifierNames[0])

# loop through all splits
for s,(train_index, test_index) in zip(TPRs.index,sss.split(COrain.iloc[:,:-1], COrain.iloc[:,-1])):
    print(s)
#     define training and testing sets
    ytrain, ytest = COrain['ls'].iloc[train_index], COrain['ls'].iloc[test_index]
    Xtrain, Xtest = COrain[COrain.columns[:-1]].iloc[train_index], COrain[COrain.columns[:-1]].iloc[test_index]
#     print(len(ytrain),len(ytest))
#     normalize training set features (column-wise)
    Xtrain_norm, Xnorm=normalize(Xtrain,axis=0,return_norm=True)
#     normalize testing set features (column-wise)
    Xtest_norm=Xtest/Xnorm
#     print(s, len(ytest[ytest==1]),sum(ytest[ytest==1].index))
    for c in range(len(classifierNames)):
```

Splitting function

Testing set (y=label, X=features)

Training set (y=label, X=features)

Split data into training and testing sets

```
#     normalize training set features (column-wise)
    Xtrain_norm, Xnorm=normalize(Xtrain,axis=0,return_norm=True)
#     normalize testing set features (column-wise)
    Xtest_norm=Xtest/Xnorm
#     print(s, len(ytest[ytest==1]),sum(ytest[ytest==1].index))
    for c in range(len(classifierNames)):
#         if c in [3,6,7,9]:continue
        if c in [3,9]:continue
#         define current classifier
        clf = classifiers[c]
#         train the classifier using the training set
        clf.fit(Xtrain_norm, ytrain)
```

Set the current model/classifier

Training proper

Train model with the training set

- K Nearest Neighbor
- Decision Tree
- Random Forest
- Ada Boost
- Gaussian Naive-Bayes

Train model with the training set

```
    for c in range(len(classifierNames)):
#         if c in [3,6,7,9]:continue
        if c in [3,9]:continue
#         define current classifier
        clf = classifiers[c]
#         train the classifier using the training set
        clf.fit(Xtrain_norm, ytrain)
#         predict values using the trained classifier and the test set
        y_pred = clf.predict(Xtest_norm)
#         Evaluate the test
#         Define true values
        y_true = ytest
```

Generate predictions

Predicted labels

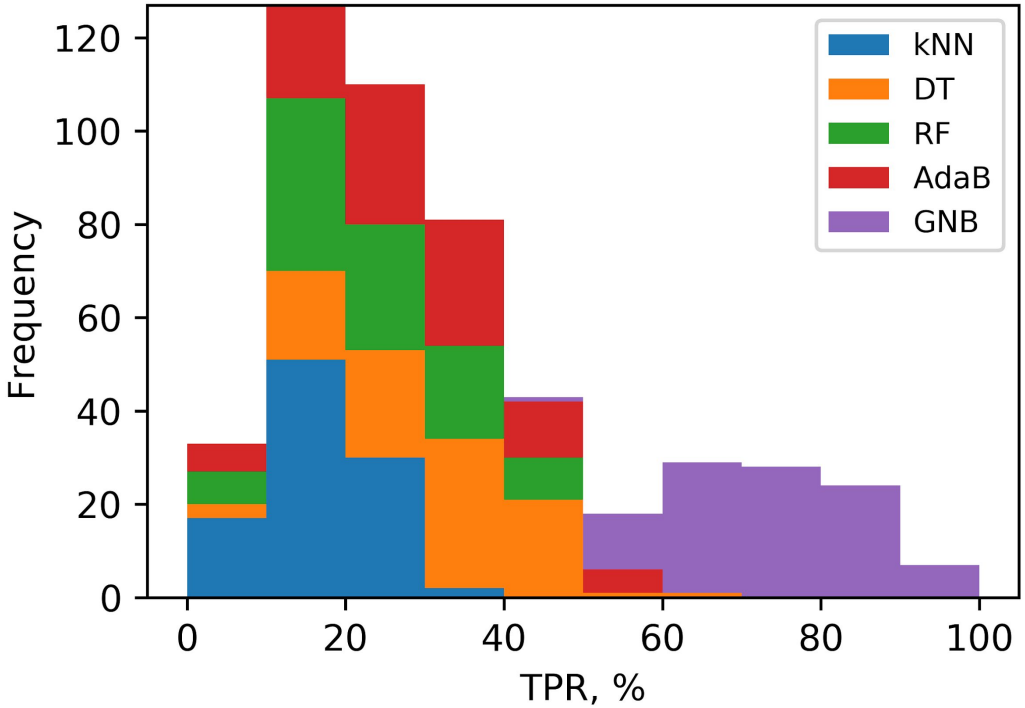
Feed testing data into model; generate predictions

```
#         predict values using the trained classifier and the test set
        y_pred = clf.predict(Xtest_norm)
#         Evaluate the test
#         Define true values
        y_true = ytest
#         Produce confusion matrix from true and predicted values
        tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
```

Function for generating confusion matrix values

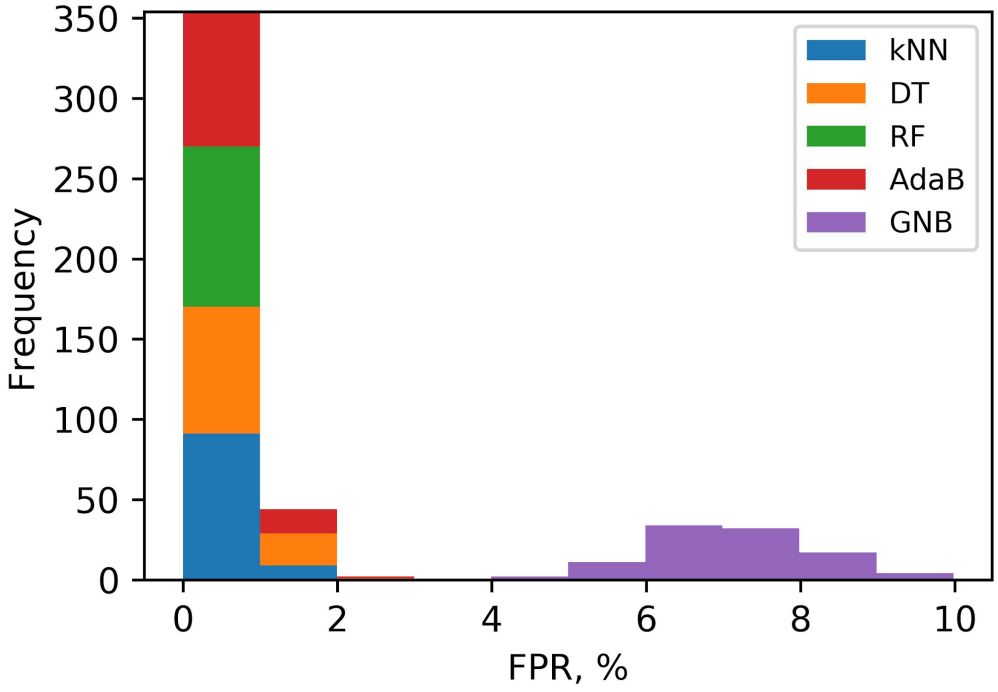
Confusion matrix: TP, FP, TN, FP

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



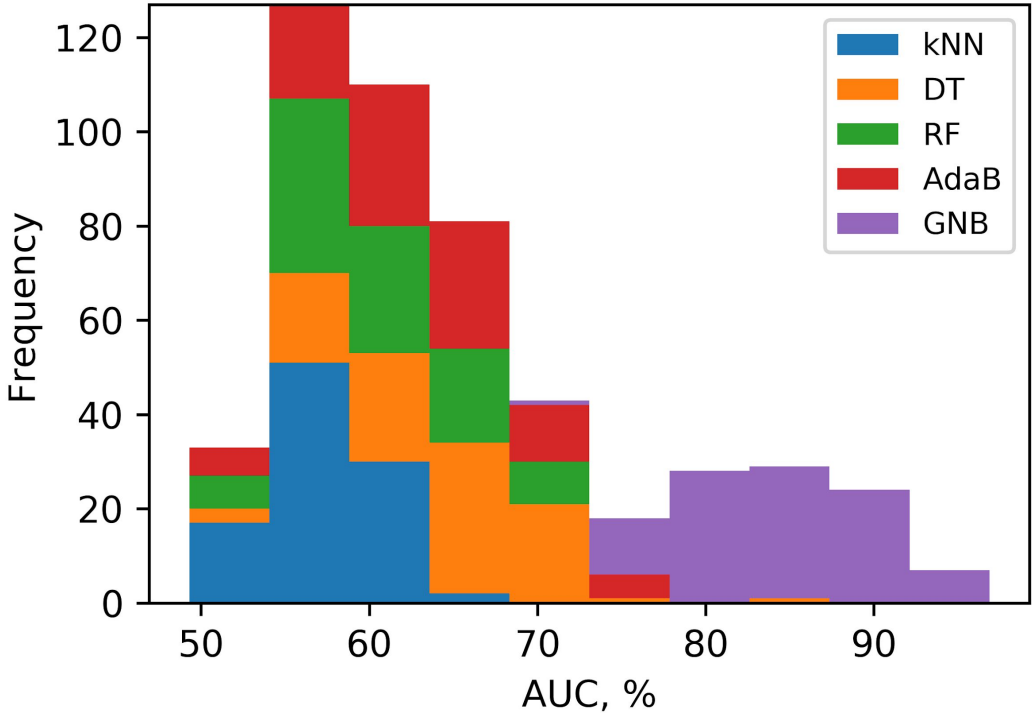
Confusion matrix: TPR (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



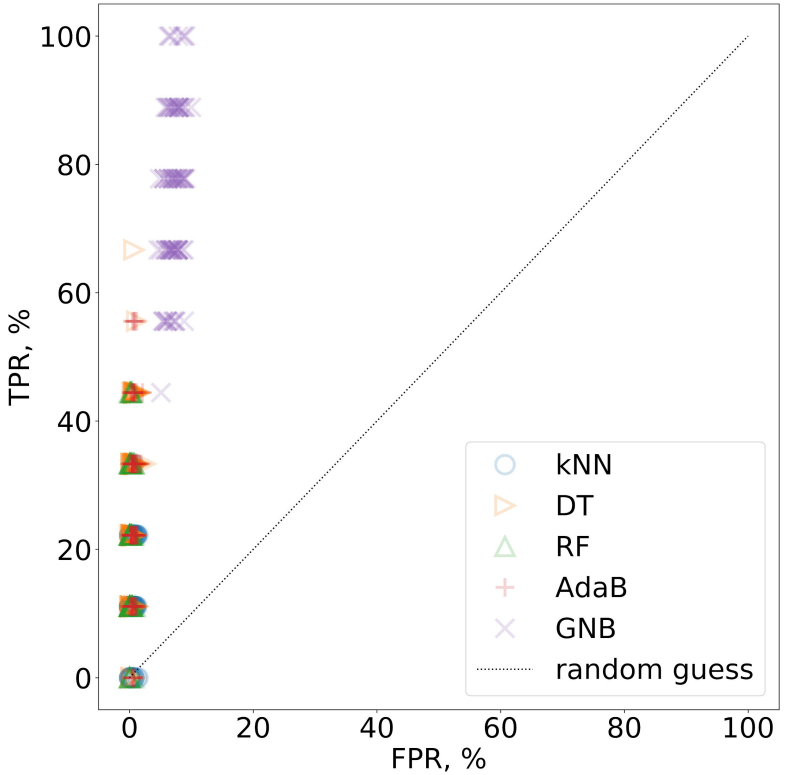
Confusion matrix: FPR (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



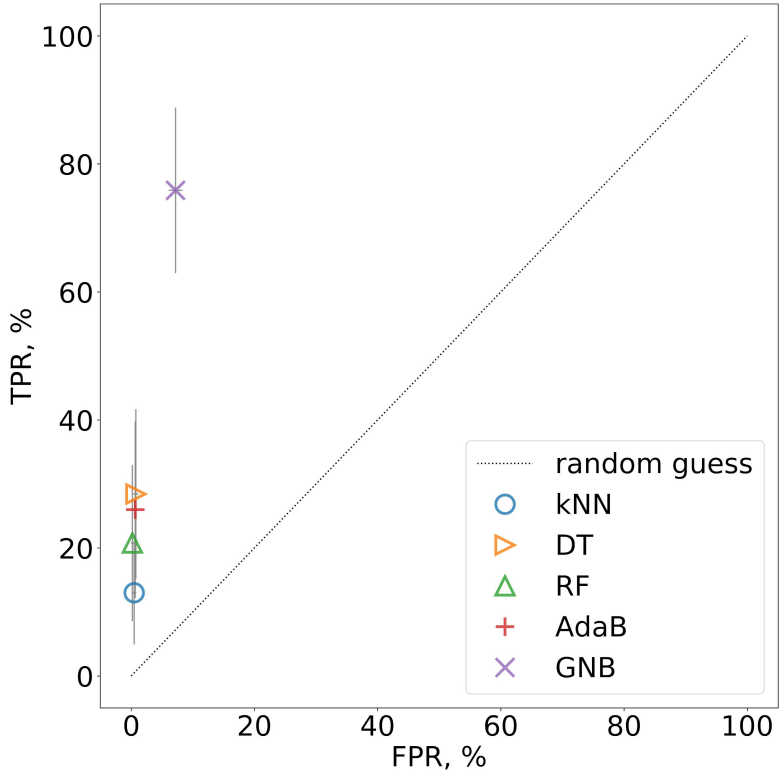
ROC curve:
area under the curve (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



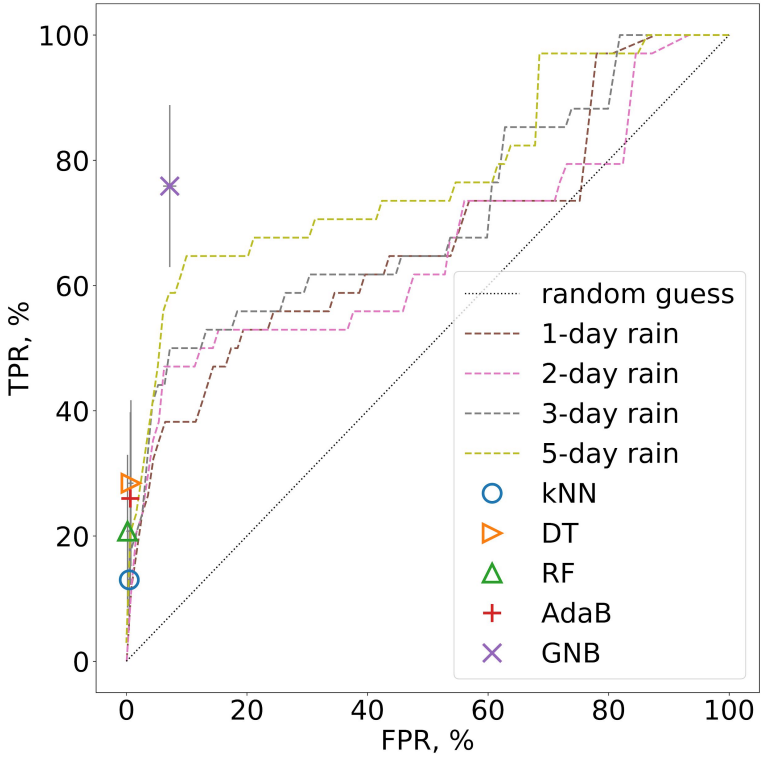
ROC curve:
area under the curve (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



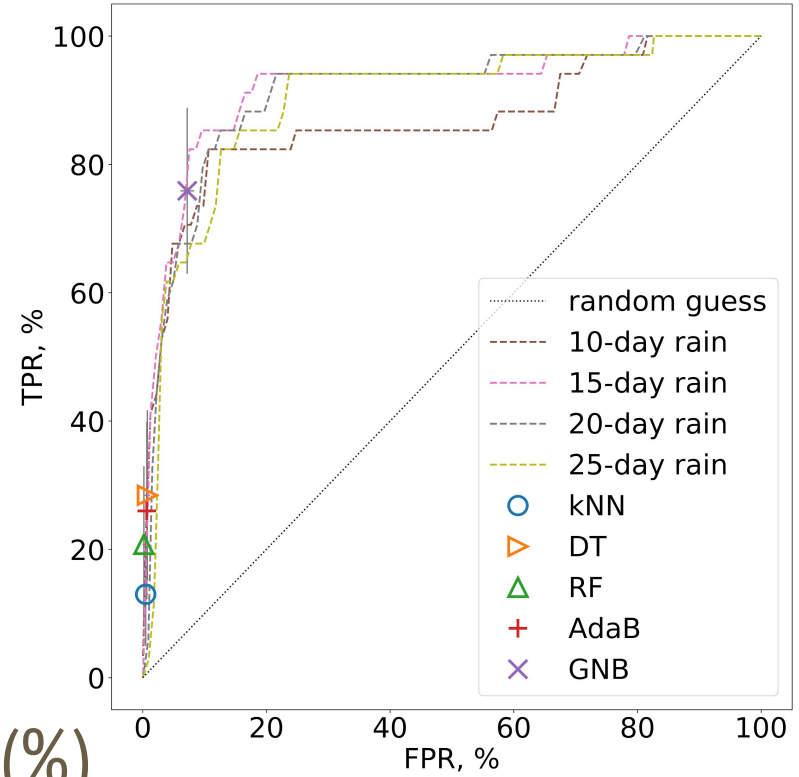
ROC curve:
area under the curve (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



ROC curve:
area under the curve (%)

Data acquisition and cleaning | Feature generation | Model training | Model testing | **Model evaluation**



ROC curve: area under the curve (%)

Preliminary results

- Gaussian Naive-Bayes
  - best ML model (TPR, AUC)
  - similar performance to 15-day cumulative rain
  - Generally outperforms other cumulative rain thresholds
- Often-used and -cited 1-day rainfall threshold performs poorly

Preliminary results

- ML models can still be improved
  - More data (rows)
  - More (or less) features (columns)
  - Optimize parameters of the individual models
  - Explore other models, or their combinations

Preliminary results

- **Potential of ML for maximizing operational data produced by agencies**
- **Cost-effective, sustainable system developed with FOSS, and potentially, by a community of app developers and researchers.**

Challenges

- Machine learning as data driven method
  - Data consistency
  - Data completeness
- Landslides as episodic phenomena
  - Relatively rare compared to non-landslide events
  - Reliable ML models require significant number of landslide events

Challenges

- Size of area of responsibility (scope of model)
  - Too small → too few landslide events
  - Too big → difficult to answer “where”

Ways forward

- **DPWH as immediate, potential end-user**
  - Rainfall data - automated access from PAGASA
  - Landslide data - improvement of system for gathering and reporting landslide data
  - Further model refinements
  - Integrate results with susceptibility/hazard maps
  - Develop probabilistic expressions of landslide prediction
  - Develop front-end for end-users

Ways forward

- Explore other potential end-users
  - With operational capability to produce quality landslide inventories
  - Willing to test the system
  - **MGB Regional Offices?**
  - **Local DRRMOs / Engineering Offices?**
  - **SUCs?**

Ways forward

- **GeoRiskPH ?**
  - Centralizing area-specific landslide inventories
  - Providing standards for data collection
  - Hosting the prediction system for operational use
- **DOST / Philippine Space Agency ?**
  - enhancing landslide inventories

References

- Agoot, Liza. “Incessant Heavy Rains Cause Landslides, Riprap Collapse in Cordillera.” Philippine News Agency RSS, Philippine News Agency, 15 Aug. 2018, www.pna.gov.ph/articles/1044915
- Buitinck, Lars et al., API design for machine learning software: experiences from the scikit-learn project, (2013) from https://arxiv.org/abs/1309.0238.
- Beguería, Santiago. "Validation and evaluation of predictive models in hazard assessment and risk management." Natural Hazards 37.3 (2006): 315-329.
- Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.
- Mueller, Andreas and Guillaume LeMaitre. Machine Learning with Scikit-Learn, Part 1|SciPy 2018 Tutorial. (July 26 2018). Retrieved July 21 2019 from https://www.youtube.com/watch?v=4PXAztQtoTg
- Nieva-Nishimori, Aleta. “‘Disaster imagination’ to help prepare communities for disaster: Solidum.” ABS-CBN News, 18 Sep. 2018. Online. Internet. 10 Jul. 2019. . Available: https://news.abs-cbn.com/news/09/18/18/disaster-imagination-to-help-prepare-communities-for-disaster-solidum
- Nolasco-Javier, Dymphna, Kumar et al. "Rapid appraisal of rainfall threshold and selected landslides in Baguio, Philippines." *Natural hazards* 78.3 (2015): 1587-1607.
- Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

Thank you.