On The Problem of Software Quality In Machine Learning Systems

Haftamu Hailu Tefera| 21.11.2022

Advisors: Juan Soto | Prof. Dr. Volker Markl | Prof. Dr. Odej Kao

Agenda

1. Motivation
2. Problem Statement
3. Solution Approach
4. Experiments
5. Conclusion

1

Motivation

Motivation

- Machine learning systems are deployed everywhere
- A ML bug refers to any imperfection that causes a inconsistency between existing and required conditions
- ML testing refers to any set of activities designed to reveal bugs

Real World Facts

Although there is testing underway in the development of ML systems, there are still problems

The AI Incident Database [1]

Common Vulnerabilities and Exposures [2]

THE RISKS DIGEST

Forum on Risks to the Public in Computers and Related Systems

2

Problem Statement

Problem Statement

How can users of machine learning systems, including practitioners and researchers be assured that Machine learning software is bug free?

3

Solution Approach

Solution Approach

1 | 2 | 3

Study the past

- Stack Overflow posts
- ML and DL systems

Familiarization

- Static analysis
- Smoke testing
- Metamorphic testing
- Model based testing

Experiments

- Reproducibility
- Seeking to discover new bugs

ML Systems Bug Analysis Study

We studied and analyzed frequent bugs, their root causes, and their impact [1,7].

Frequent Bugs

- Data bugs
- Coding bugs
- Structural bugs

- Type/shape mismatch
- Incorrect data structure
- Incorrect parameter init.

Root Causes

- Incorrect model parameters
- Inter API compatibility
- Computation model confusion

- Model parameters
- Suboptimal hyperparameters
- Inconsistency b/n libraries

Bugs Impacts

- Poor performance
- Program crash
- Incorrect functionality

Model training
Data preparation

Terminologies

- **Static analysis** is the analysis of software programs performed without executing them
- **Smoke tests** assert that the most crucial functions of a program work
- **Metamorphic Testing** is founded on the principle of if the correct output for the input is not known, we can validate or verify the system based on the outputs of multiple related inputs
- **Model Based Testing** is a method where we test a piece of software based on its expected input and output.

4

Experiments

Experiment Design

1. Experiment class 1: Static analysis
2. Experiment class 2 : Smoke testing
3. Experiment class 3 : Model based testing
4. Experiment class 4 : Metamorphic testing
5. Experiment class 5 : Evaluation of classification algorithms

Static Analysis

We analyzed open-source ML software repositories with SonarCloud

Scikit-learn	• Missing argument • Unexpected argument • Unreachable code
TensorFlow	• Method call error • File not found • Empty function body
Keras|PyTorch	• Unused variable or argument • Code duplication with literals • Image version tagging

Smoke Testing

Herbold, Steffen. "Smoke testing for machine learning: simple tests to discover severe bugs"

Scikit-learn

- Decision Tree
- Random Forest
- SVM | SGD | KNN
- Naive Bayes

- Extreme class level imbalance
- All data from the same class
- Extremely large values
- Attribute reference error

Spark ML

- Decision Tree
- Random Forest
- Naive Bayes
- Logistic Registic

- Unsupported arguments
- Single training instance for a class
  - Division by zero (std)
- Too many distinct categories

Model Based Testing

A sentiment analysis model evaluated on Google Cloud NLP System

```
Please wait as we prepare the table data...
```

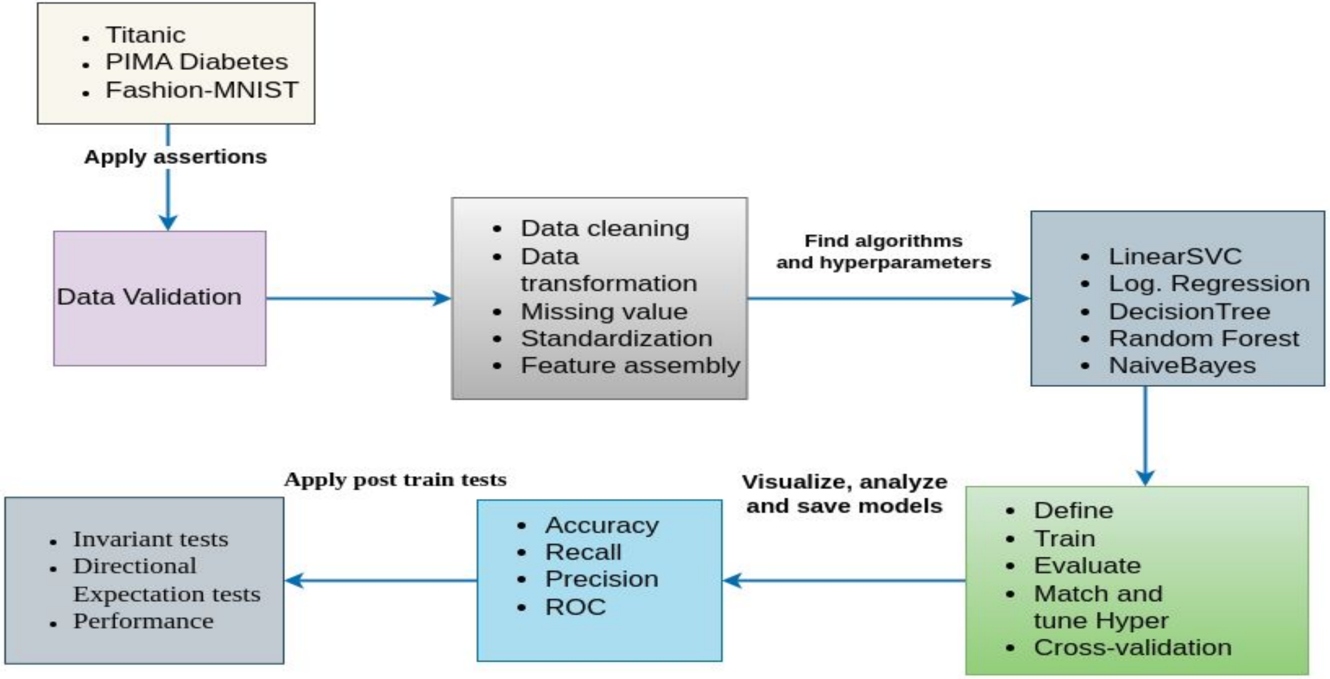
	Capabilities	**M**inimum **F**unctionality **T**est *failure rate % (over N tests)*	**INV**ariance Test *failure rate % (over N tests)*	**DIR**ectional Expectation Test *failure rate % (over N tests)*
+	Vocabulary	48.6% (5)	16.2% (1)	34.6% (4)
+	Robustness		13.6% (5)	
+	NER		20.8% (3)	
+	Fairness		1.6% (4)	
+	Temporal	36.6% (1)		2.0% (1)
+	Negation	100.0% (9)		
+	SRL	90.8% (5)		

Evaluation of Classification Algorithms

We evaluated classification algorithms from Spark ML, Scikit-learn and Keras network to detect or minimize bugs at an early stage.

Based on model behavioral testing for NLP models, we evaluated Scikit-learn models using invariant and directional expectation tests.

Methodology



Scikit-learn

Spark ML

Keras

Jupyter Notebook

Python

Pandas DataFrame

Spark DataFrame

Apply Pre-train Tests to Improve Data Quality

- To avoid unexpected column(s) , feature values and label values
- Ensure the range of values are in the specified range

```
assert set(titanic["Sex"].unique()) == set(("female", "male")), "Unknown gender"
assert set(titanic["Survived"].unique()) == set((0, 1)), "Unknown survived value"
assert set(titanic["Embarked"].unique()) == set(("S", "C", "Q"), "Unknown Embarked"
assert set(titanic["Pclass"].unique()) == set((1, 2, 3)), "Unknown Pclass value"
assert titanic["Age"].min() >= 0, "Age should be positive"
assert titanic["SibSp"].min() >= 0, "SibSp should be positive"
assert titanic["Fare"].min() >= 0, "Fare should be positive"
assert titanic["Parch"].min() >= 0, "Parch should be positive"
assert titanic["Parch"].max() <= 6, "Parch should be less than 7"
```

Data Preprocessing and Feature Engineering

Missing value	Duplicate	Standardization	Encoding	Sampling	Vectorization

```
round(titanic["Age"][titanic["Name"].str.contains("Mrs.")
& titanic["Sex"].str.contains("female")].mean())
titanic["Embarked"][titanic["Embarked"].isna()] = "S"
titanic.drop(index=titanic[titanic.drop(columns="Survived").duplicated()].index, inplace=True)
titanic["Sex"] = np.where(titanic["Sex"] == "male", 1, 0)
titanic = pd.concat([titanic[titanic['Survived'] == 0]
    .sample(len(titanic[titanic['Survived'] == 1]), random_state=seed),
titanic[titanic['Survived'] == 1]],axis=0)

titanic = titanic.rename(columns={'Survived': 'label'})
va = VectorAssembler(inputCols = col, outputCol='features')
va_df = va.transform(data)
va_df = va_df.select(['features', 'label'])
return va_df.randomSplit([1-test_size, test_size], seed=seed)
```

Pandas

Spark

Define and Match Hyperparameter

```
grid = {"LinearSVC": {"max_iter": [10, 50, 100, 500],"tol":
[1e-2, 1e-3, 1e-4, 1e-5], "C": [0.1, 0.01, 0.001, 0.0001],
"fit_intercept": [True, False]},
    "LogisticRegression": {"max_iter": [10, 50, 100, 500],
    "tol": [1e-2, 1e-3, 1e-4, 1e-5],"C": [0.1, 0.01, 0.001, 0.0001],
    "fit_intercept": [True, False]},
```

Scikit-learn hyperparameters

Match hyperparameter names

```
para = { "LinearSVC": {"max_iter": ["maxIter"],"tol": ["tol"],"C": ["regParam"],
      "fit_intercept": ["fitIntercept"]},
    "LogisticRegression": {"max_iter": ["maxIter"],"tol": ["tol"],"C": ["regParam"],
    "fit_intercept": ["fitIntercept"]},
```

PySpark hyperparameters

4. Mapping Algorithm Names

```
spark_grid = {}
spark_model_name_mapping = {
    "GaussianNB": "NaiveBayes",
    "GradientBoostingClassifier": "GBTClassifier",
    "MLPClassifier": "MultilayerPerceptronClassifier",
    "OneVsRestClassifier": "OneVsRest"}

for sk_model_name, param_mapping in para.items():
    spark_model_name = sk_model_name
    if sk_model_name in spark_model_name_mapping.keys():
        spark_model_name = spark_model_name_mapping.get(sk_model_name)
    spark_grid[spark_model_name] = {}
    for skt_name, pyspark_name in param_mapping.items():
        spark_grid[spark_model_name][
            pyspark_name[0]] = grid[sk_model_name][skt_name]
grid = spark_grid
```

(Scikit-learn: PySpark)

There are naming variations

Model Training and Evaluation : Scikit-learn

Define the models (1) Train the models (2) Evaluate the models (2)

```
sklearn_classifiers = [
    sklearn_models.LinearSVC(random_state=seed),
    sklearn_models.LogisticRegression(random_state=seed),

for clf in sklearn_classifiers:
    clf.fit(x_train, y_train)
    accuracy, confusion, roc, precision, recall = eval_methods.eval(
        clf, y_test, clf.predict(x_test), ax, bx)
```

Model Training and Evaluation : PySpark

```
pyspark_classifiers = [
    pyspark_models.LinearSVC(labelCol="label"),
    pyspark_models.LogisticRegression(labelCol="label"),

for clf in pyspark_classifiers:
    clf = clf.fit(train)
    real = np.array([
        1 if "1" in str(x) else 0
        for x in clf.transform(test).select("label").collect()])
    pred = np.array([
        1 if "1" in str(x) else 0
        for x in clf.transform(test).select("prediction").collect()])
    accuracy, confusion, roc, precision, recall = eval_methods.eval(
        clf, real, pred, ax, bx)
```

Define models

Train models

Evaluate models

Apply Post-train Test to Ensure Expected Learned Behavior

```python
path = "./models/titanic/sklearn/"
for i in os.listdir(path):
    model = pickle.load(open(path+i, 'rb'))
    X = datasets.get_titanic().iloc[291]
    y = X["Survived"]
    X = X[1:]
    p2_prob = model.predict(np.array(X).reshape(1, -1))[0] #1.0
    X['Embarked'] = 2.47593535
    assert p2_prob == model.predict(np.array(X).reshape(1, -1))[0]#1.0
    p2_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 1.0
    X['Sex'] = 0.83739228
    p2_male_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.56
    assert p2_prob > p2_male_prob,
    'Changing gender from female to male should decrease survival probability.
    X['Pclass'] = 0.95828974
    p2_class_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.0
    assert p2_prob > p2_class_prob,
    'Changing class from 1 to 3 should decrease survival probability.'
```

Pclass, Sex and Fare are relevant features

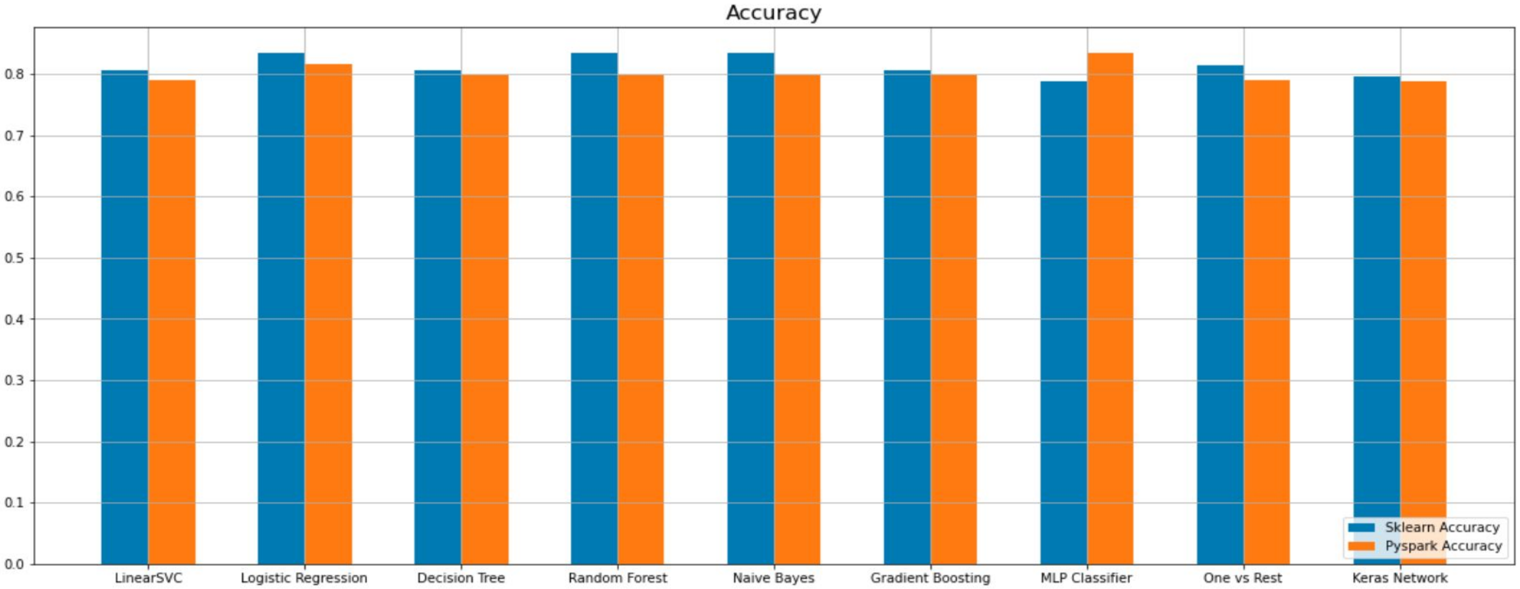
```python
titanic = datasets.get_titanic()
titanic.iloc[291]
```

```
Survived     1.000000
Pclass      -1.371412
Sex         -1.201153
Age          0.601520
SibSp       -0.470631
Parch       -0.428538
Ticket      -0.753102
Fare         0.715063
Embarked    -0.610667
Name: 61, dtype: float64
```

Evaluation

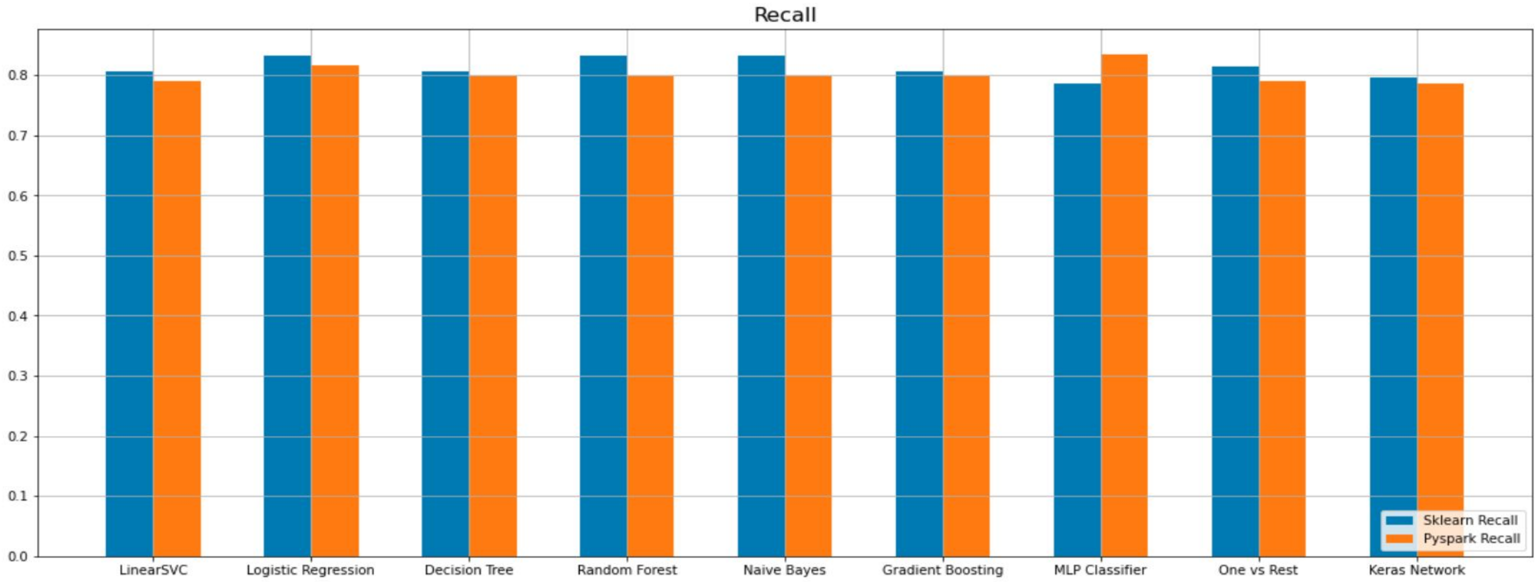
Accuracy

The ratio of correctly predicted instances to the total observations



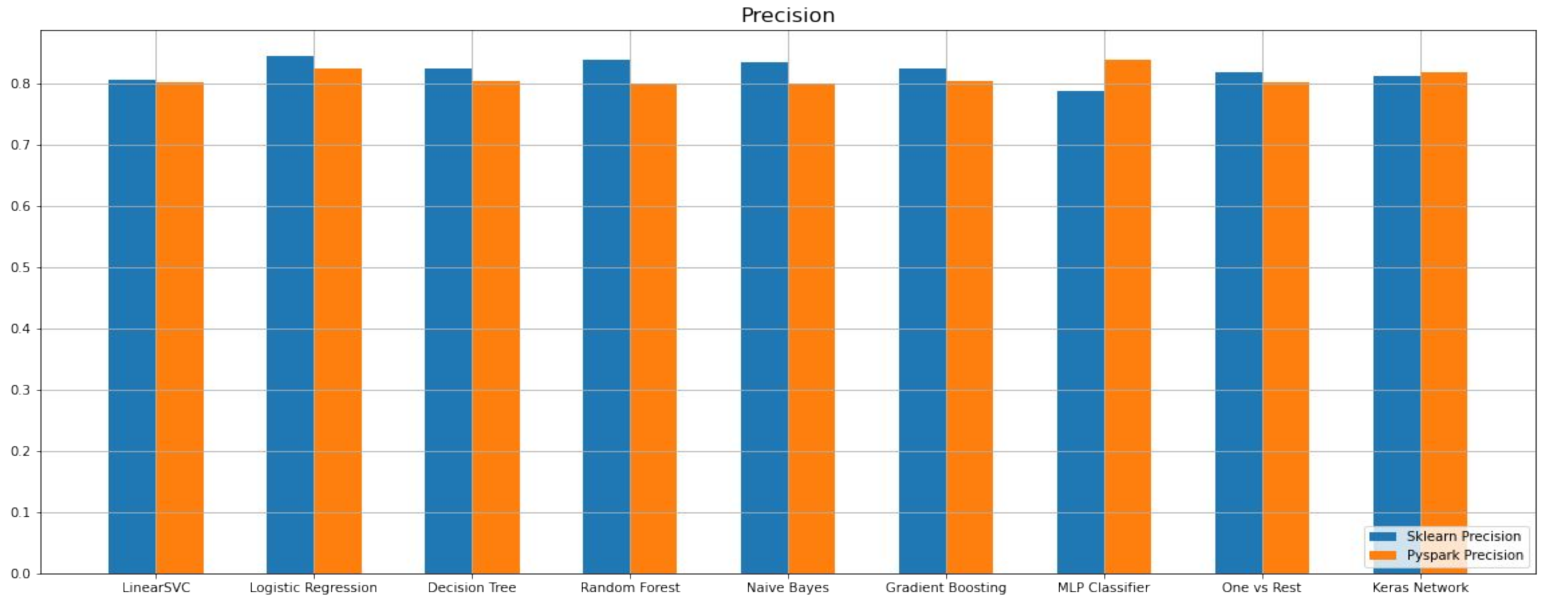
Recall

The ratio of r**elevant instances** that has been retrieved over the total number of instances



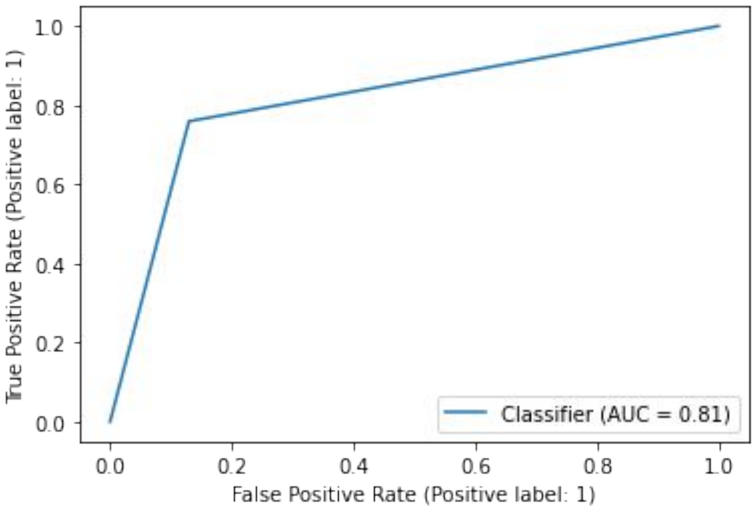
Precision

The fraction of **survived instances** among the r**etrieved instances** (Survived + False Positives)

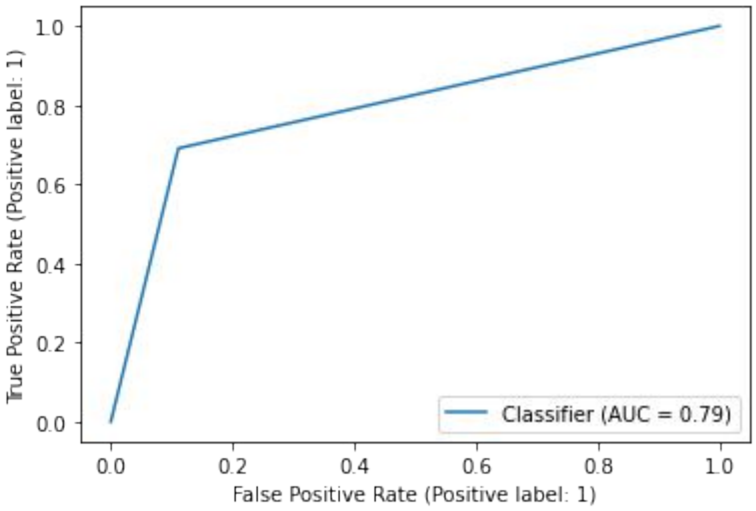


ROC for Linear SVC Models

The LinearSVC from scikit-learn better identifies the survived passengers



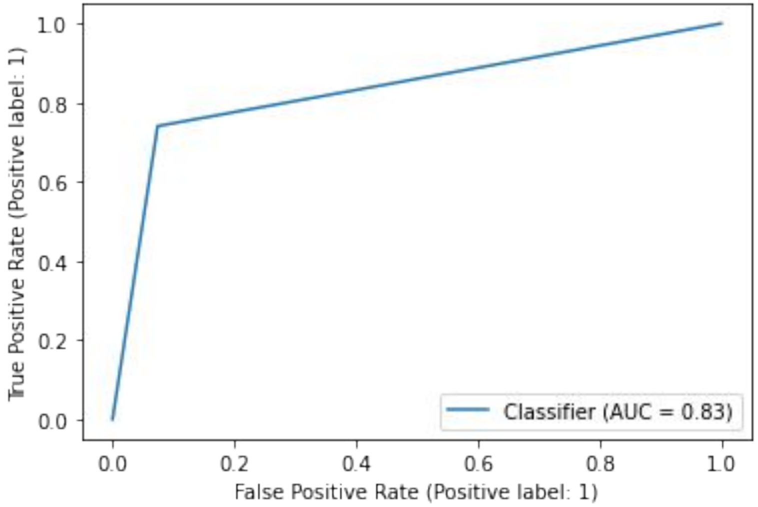
a) Scikit-learn



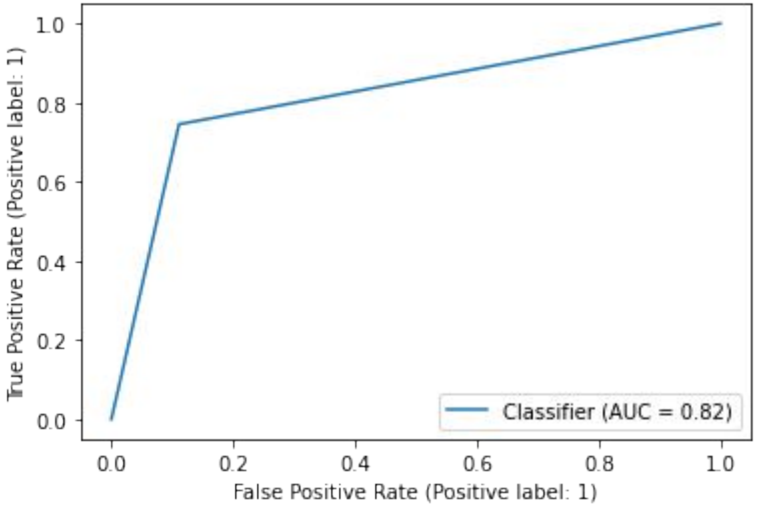
b) PySpark

ROC of Logistic Regression

The two logistic regressions identifies the survived passengers almost equally



a) Scikit-learn



b) PySpark

Directional Expectation Tests for LinearSVC

The LinearSVC scikit-learn model did not pass the test

```
Testing model: LinearSVC(C=0.0001, max_iter=10, random_state=0, tol=0.01)

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
/tmp/ipykernel_38917/3038047054.py in <cell line: 2>()
      9     X['Sex'] = 0.83739228  #Change gender
     10     p2_male_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.56
---> 11     assert p2_prob > p2_male_prob, 'Changing gender from female to male should decrease survival probab
ility.'
     12     X['Pclass'] = 0.95828974 # Change class
     13     p2_class_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.0

AssertionError: Changing gender from female to male should decrease survival probability.
```

Change the gender value from **-1.2011** to **0.8373**

Directional Expectation Tests for Gradient Boosting Classifier

The Gradient Boosting Classifier model did not pass the test

```
Testing model: GradientBoostingClassifier(learning_rate=0.01, max_depth=2, max_features='auto',
                           random_state=0)

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
/tmp/ipykernel_38917/1720687962.py in <cell line: 2>()
     12     X['Pclass'] = 0.95828974 # Change class
     13     p2_class_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.0
---> 14     assert p2_prob > p2_class_prob, 'Changing class from 1 to 3 should decrease survival probability.'
     15     X['Fare'] = -0.575978 # # Lower fare
     16     p2_fare_prob = model.predict(np.array(X).reshape(1, -1))[0]  # 0.85

AssertionError: Changing class from 1 to 3 should decrease survival probability.
```

Change passenger class from **-1.37** to **0.9582**

5

Conclusion

Summary

- We study the past and perform static analysis
- We examined and experimented with three novel testing techniques
- Unable to detect new bugs and the performance difference is negligible
- We implemented pre-train and post train tests tests to improve data quality and ensure model learned behavior
- GBTClassifier and LinearSVC from PySpark supports binary classification

Future Work

- Extend to model post-train testing for PySpark models
- Experiment tracking for best hyperparameters, performance scores, visualizations and other model artifacts
- Provide holistic benchmark with state-of-the-art by providing algorithms specific capabilities
- Apply Metamorphic testing and smoke testing to classification algorithms

References

1. AI Incident Database
2. CVE.org
3. Islam, Md Johirul, et al. "What do developers ask about ml libraries? a large-scale study using stack overflow." *arXiv preprint arXiv:1906.11940* (2019).
4. Ribeiro, Marco Tulio, et al. "Beyond accuracy: Behavioral testing of NLP models with CheckList." *arXiv preprint arXiv:2005.04118* (2020).
5. Zhang, Jie M., et al. "Machine learning testing: Survey, landscapes and horizons." *IEEE Transactions on Software Engineering* (2020).
6. Breck, Eric, et al. "The ML test score: A rubric for ML production readiness and technical debt reduction." *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017.
7. Humbatova, Nargiz, et al. "Taxonomy of real faults in deep learning systems." *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.