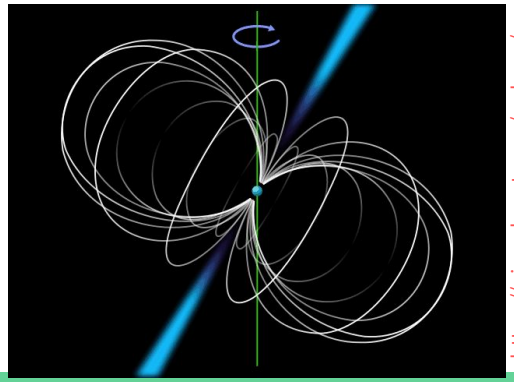
Pulsar Classification

Sam Johnson and Corwin Schmidt

Goal

- Determine best learning algorithm to classify radio emission sources as either pulsars or noise/RFI (Radio Frequency Interference)
- Pulsars: a rare type of Neutron star
- Pulsar rotations cause periodic broadband radio emissions detectable on Earth
- Each pulsar's emission pattern is unique, i.e. a model is required to distinguish them from other noise
    - 700+ rotation/sec to ~0.5 rotation/min



https://nineplanets.org/pulsars/

Significance

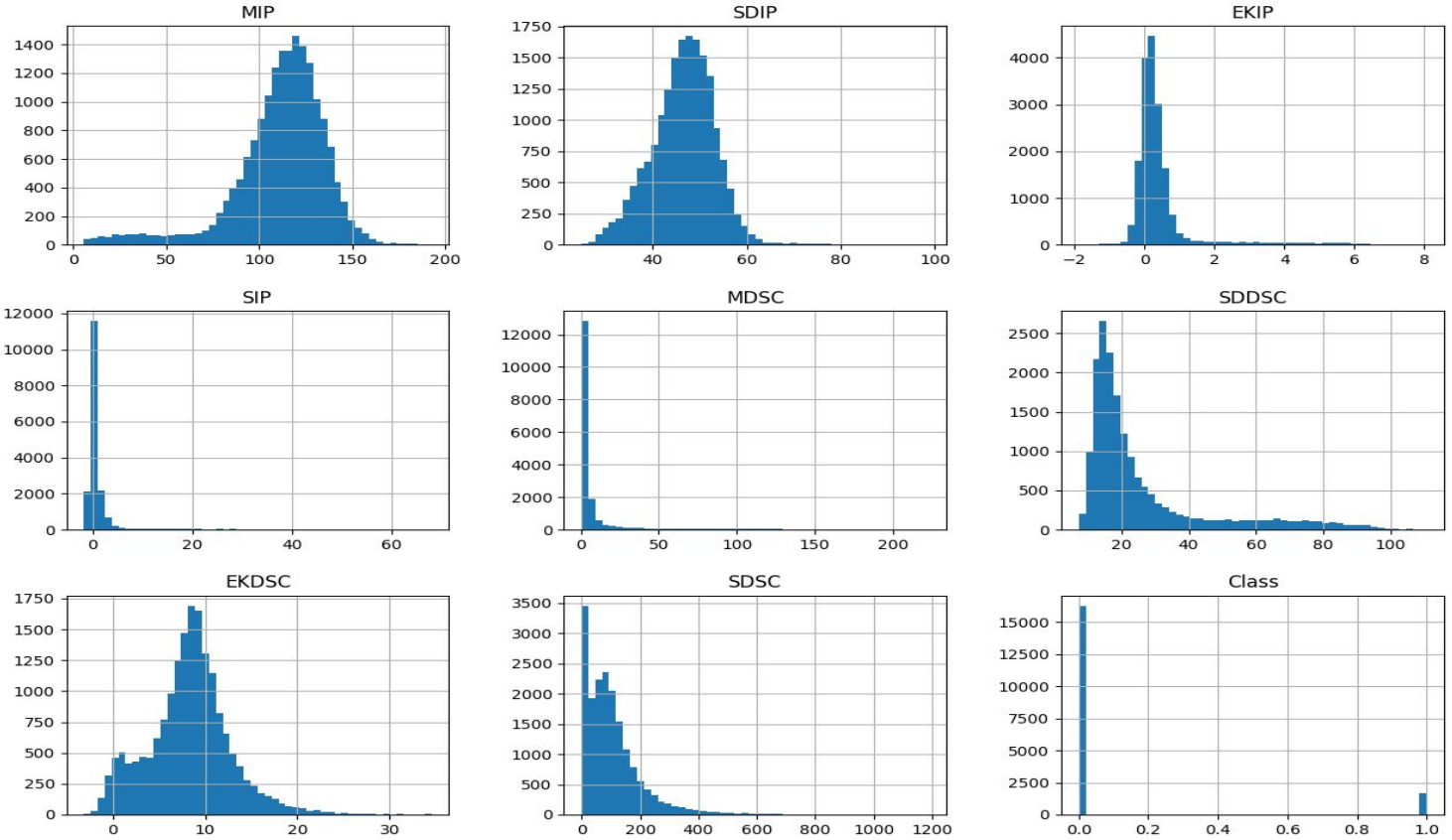
- Pulsar orbiting another star affects regularity of frequency transmitted
  - Masses of both
  - Distance between the two
- Massive object intersecting pulsar radio waves affects frequency received
  - Mass of intersecting object
  - Trajectory
- This allows us to...
  - Investigate the interstellar medium
  - Detect gravitational waves
    - Locate extrasolar planets in orbit

https://nineplanets.org/pulsars/

Data

- HTRU2 Data Set: 17,898 total examples, of which 1,639 are positive and 16,259 are negative.
    - R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656
- 8 continuous variables and 1 class variable:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class


MIP
SDIP
EKIP
SIP
MDSC
SDDSC
EKDSC
SDSC
Class

Data Preparation

```
# Automatically Importing All Classifiers

estimators = all_estimators(type_filter='classifier')

all_class = []
all_class_names = []
for name, Classifiers in estimators:
    try:
        if name != 'GaussianProcessClassifier' and name != 'DummyClassifier':
            print('Appending', name)
            reg = Classifiers()
            all_class.append(reg)
            all_class_names.append(name)
    except Exception as e:
        print(e)

print(all_class)
print(all_class_names)

# Load and Describe Data

def load_pulsar_data():
    csv_path = os.path.abspath("HTRU_2.csv")
    return pd.read_csv(csv_path)

pulsar = load_pulsar_data()
print(pulsar.describe())

print(pulsar.corr())
print('*'*100)

```

```
# Train/Test Split and Preprocess Data
pulsar["EKIP_cat"] = pd.cut(pulsar["EKIP"],bins=[-2.0,0.027098,0.223240,0.473325,np.inf],labels=[1,2,3,4],right=True)

split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index, test_index in split.split(pulsar,pulsar["EKIP_cat"]):
    train_set = pulsar.loc[train_index]
    test_set = pulsar.loc[test_index]

for set_ in(train_set,test_set):
    set_.drop("EKIP_cat",axis=1,inplace=True)

ptrain = train_set.copy()
ptest = test_set.copy()

ptrain_attrib = ptrain.drop("Class",axis=1)
ptrain_labels = ptrain["Class"].copy()
ptest_attrib = ptest.drop("Class",axis=1)
ptest_labels = ptest["Class"].copy()

scaler = StandardScaler()
ptrain_attrib = scaler.fit_transform(ptrain_attrib)
ptest_attrib = scaler.fit_transform(ptest_attrib)

pulsar.hist(bins=50, figsize=(15,11))
plt.savefig('variables.png')

```

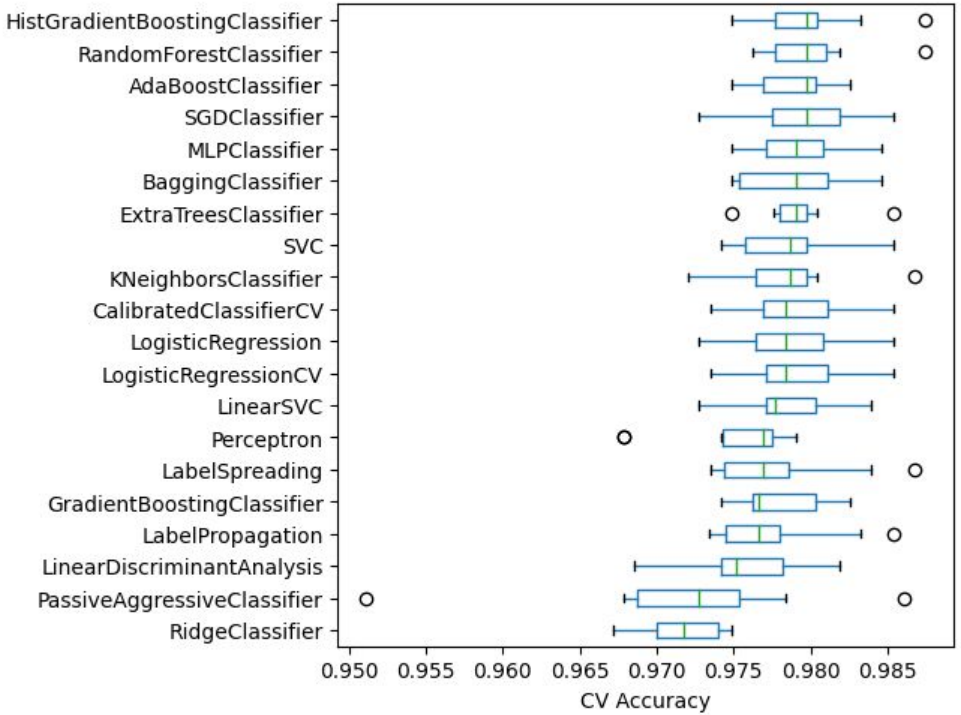
- scikit-learn
- Imported all classifiers
- Stratified shuffle split
- Leave *k* out

Model Training

```
80  def comparison(models,model_names):
81      cv_data = []
82      errors = []
83      passed_models = []
84      for i in range(len(models)):
85          x = run(models[i])
86          if type(x) == dict:
87              cv_data += [x]
88          else:
89              errors += [models[i]]
90      for j in range(len(models)):
91          if models[j] not in errors:
92              passed_models += [model_names[j]]
93      figs = [test_best(cv_data, passed_models), box_acc(cv_data, passed_models), box_prec(cv_data, passed_models), box_r
94      for k in range(len(figs)):
95          figs[k].savefig(f'fig_{k}.png',bbox_inches='tight')
96      return test_best(cv_data, passed_models)
97
98  def run(model):
99      print(f"checking {model}")
100     try:
101         cv_outer = KFold(n_splits=10, shuffle=True, random_state=2)
102         cv_output_dict = cross_validate(model, ptrain_attrib, ptrain_labels, scoring=["accuracy","precision","recall"],
103         return cv_output_dict
104     except:
105         pass
```

- *k*-fold Cross-validation
- 30+ classifiers

Model Evaluation



- Performance on training data
- Accuracy = # of correct predictions / total # of predictions

Model Evaluation

	Accuracy	Precision	Recall
MLPClassifier	0.9813	0.9444	0.8421
HistGradientBoostingClassifier	0.9802	0.9375	0.8359
SVC	0.9796	0.9529	0.8142
SGDClassifier	0.9796	0.9562	0.8111
LogisticRegressionCV	0.9793	0.9527	0.8111
LogisticRegression	0.9793	0.9527	0.8111
BaggingClassifier	0.9791	0.9247	0.8359
CalibratedClassifierCV	0.9791	0.9526	0.808
RandomForestClassifier	0.9791	0.9247	0.8359
ExtraTreesClassifier	0.9791	0.9336	0.8266
LinearSVC	0.9785	0.9522	0.8019
KNeighborsClassifier	0.9782	0.9391	0.8111
PassiveAggressiveClassifier	0.9774	0.9764	0.7678
GradientBoostingClassifier	0.9771	0.8977	0.8421
AdaBoostClassifier	0.9768	0.9348	0.7988
LabelSpreading	0.9757	0.9275	0.7926
LabelPropagation	0.9749	0.9206	0.7895
LinearDiscriminantAnalysis	0.9732	0.9486	0.743
Perceptron	0.9723	0.8636	0.8235
RidgeClassifier	0.9704	0.9738	0.6904
RidgeClassifierCV	0.9704	0.9738	0.6904
DecisionTreeClassifier	0.9684	0.8261	0.8235
NearestCentroid	0.9676	0.8581	0.7678
ExtraTreeClassifier	0.967	0.8213	0.8111
QuadraticDiscriminantAnalysis	0.9665	0.7892	0.8576
RadiusNeighborsClassifier	0.9665	0.7892	0.8576
BernoulliNB	0.9522	0.6854	0.87
GaussianNB	0.943	0.64	0.8421

- Performance on test data
- Top classifier: Multi-Layer Perceptron Classifier
- Precision: TP / (TP+FP)
- Recall: TP / (TP+FN)

Conclusion

- Best model: Multi-Layer Perceptron Classifier
- Potential for improvement
  - Parameter tuning
- Benefits of a performant model
  - Accurate classification = less time wasted on incorrect classification
  - Save time and money by shortening the observation period required to classify a radio emission source