Dealing With Imbalanced Classes

- Stratified Sampling
- Random Undersampling
- Random Oversampling
- Oversample Synthetic Minority Items
    - SMOTE
    - ADASYN
- Other methods

Stratified Sampling

In [3]:
```
from sklearn.model_selection import StratifiedKFold

X = np.ones(10)
y = [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

skf = StratifiedKFold(n_splits=3)
for train, test in skf.split(X, y):
    print("%s %s" % (train, test))
```

```
[2 3 6 7 8 9] [0 1 4 5]
[0 1 3 4 5 8 9] [2 6 7]
[0 1 2 4 5 6 7] [3 8 9]
```

Random Sampling

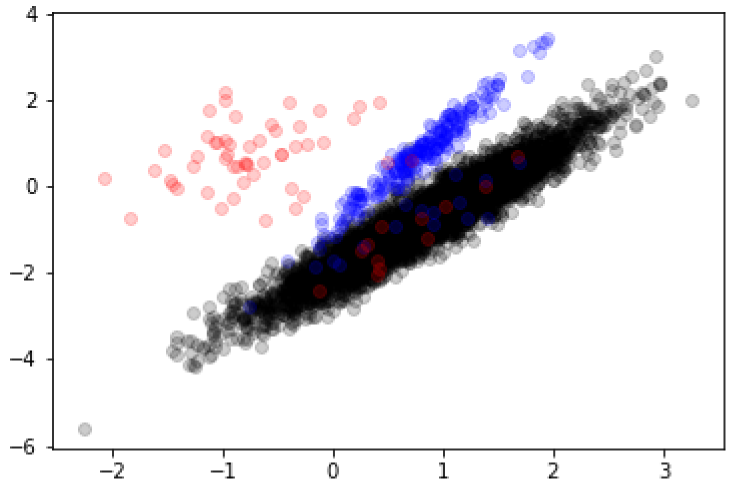
- Randomly Undersample majority class
- Randomly Oversample minority class

Example Dataset

In [4]:
```
from sklearn.datasets import make_classification
from collections import Counter
X, y = make_classification(n_samples=5000, n_features=2, n_informative=2,
                           n_redundant=0, n_repeated=0, n_classes=3,
                           n_clusters_per_class=1,
                           weights=[0.01, 0.05, 0.94],
                           class_sep=0.8, random_state=0)
Counter(y).items()
```

Out[4]:
```
dict_items([(2, 4674), (1, 262), (0, 64)])
```

In [5]:
```
plt.scatter(X[y==2,0],X[y==2,1],c='k', alpha=.2);
plt.scatter(X[y==1,0],X[y==1,1],c='b', alpha=.2);
plt.scatter(X[y==0,0],X[y==0,1],c='r', alpha=.2);
```



Using imblearn

In [6]:
```
# conda install -c conda-forge -n eods-s20 imbalanced-learn
```

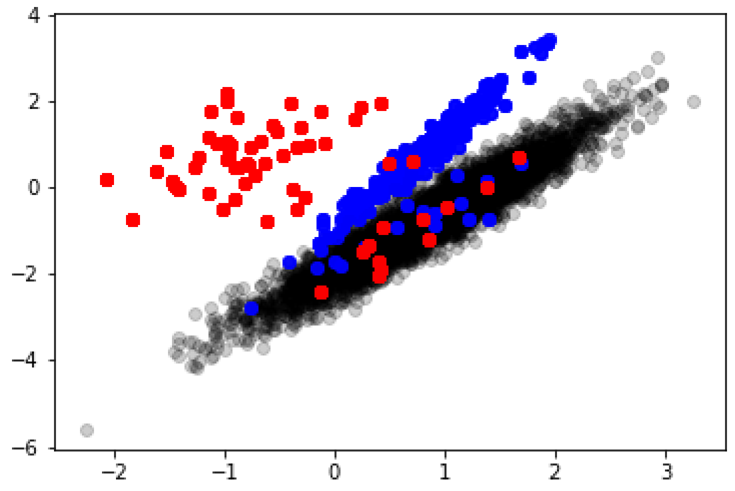
Random Oversampling

```
In [7]: from imblearn.over_sampling import RandomOverSampler

        ros = RandomOverSampler(random_state=0)
        X_r, y_r = ros.fit_sample(X, y)
        Counter(y_r).items()

Out[7]: dict_items([(2, 4674), (1, 4674), (0, 4674)])
```

```
In [8]: plt.scatter(X_r[y_r==2,0],X_r[y_r==2,1],c='k', alpha=.2);
        plt.scatter(X_r[y_r==1,0],X_r[y_r==1,1],c='b', alpha=.2);
        plt.scatter(X_r[y_r==0,0],X_r[y_r==0,1],c='r', alpha=.2);
```



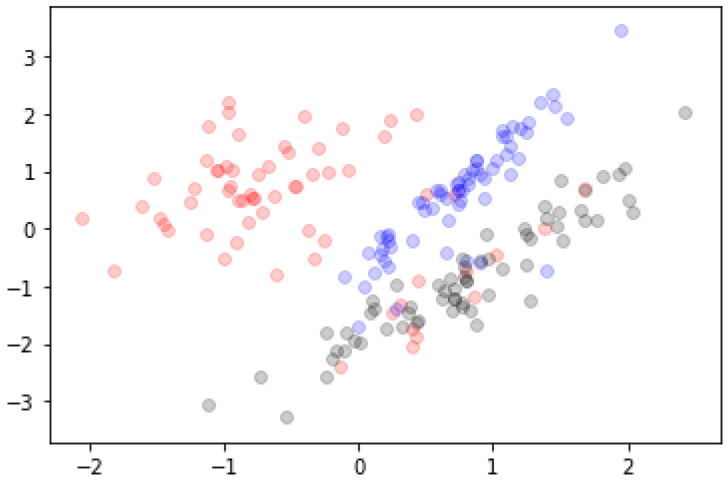
Random Undersampling

In [9]:
```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=0)
X_r, y_r, = rus.fit_sample(X, y)
Counter(y_r).items()
```

Out[9]:
```
dict_items([(0, 64), (1, 64), (2, 64)])
```

In [10]:
```
plt.scatter(X_r[y_r==0,0],X_r[y_r==0,1],c='r', alpha=.2);
plt.scatter(X_r[y_r==1,0],X_r[y_r==1,1],c='b', alpha=.2);
plt.scatter(X_r[y_r==2,0],X_r[y_r==2,1],c='k', alpha=.2);
```



Oversample Sythetic Minority Items

- SMOTE: Synthetic Minority Oversampling
- ADASYN: Adaptive Synthetic Minority Oversampling

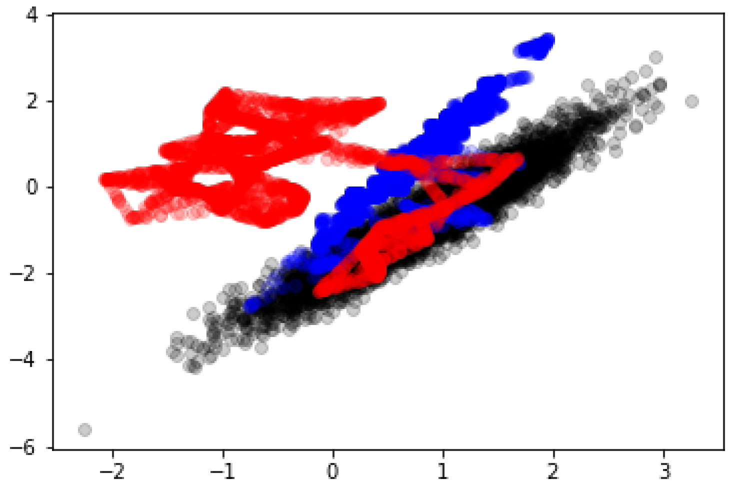
SMOTE: Synthetic Minority Oversampling

In [11]:
```
from imblearn.over_sampling import SMOTE

X_r, y_r = SMOTE().fit_sample(X, y)
Counter(y_r).items()
```

Out[11]:
```
dict_items([(2, 4674), (1, 4674), (0, 4674)])
```

In [12]:
```
plt.scatter(X_r[y_r==2,0],X_r[y_r==2,1],c='k', alpha=.2);
plt.scatter(X_r[y_r==1,0],X_r[y_r==1,1],c='b', alpha=.2);
plt.scatter(X_r[y_r==0,0],X_r[y_r==0,1],c='r', alpha=.2);
```



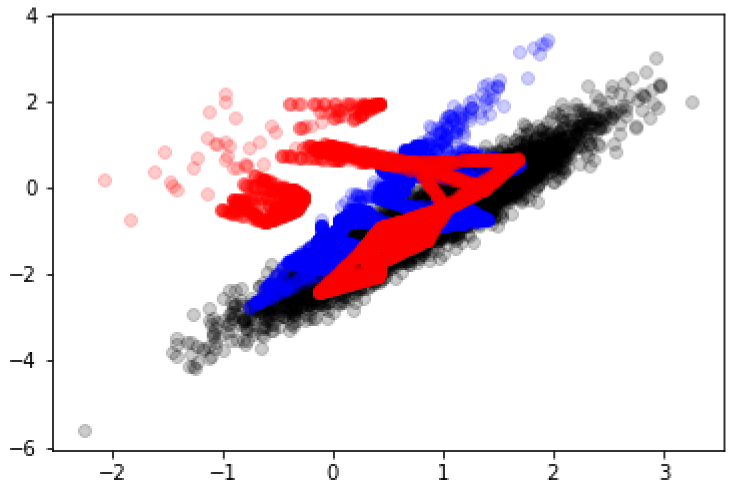
ADASYN: Adaptive Synthetic Minority Oversampling

In [13]:
```
from imblearn.over_sampling import ADASYN

X_r, y_r = ADASYN().fit_sample(X, y)
Counter(y_r).items()
```

Out[13]:
```
dict_items([(2, 4674), (1, 4662), (0, 4673)])
```

In [14]:
```
plt.scatter(X_r[y_r==2,0],X_r[y_r==2,1],c='k', alpha=.2);
plt.scatter(X_r[y_r==1,0],X_r[y_r==1,1],c='b', alpha=.2);
plt.scatter(X_r[y_r==0,0],X_r[y_r==0,1],c='r', alpha=.2);
```



Other methods for dealing with imbalanced classes

- Adjust class weight (sklearn)
- Adjust decision threshold (sklearn)
- Treat as anomaly detection
- Buy more data

See https://imbalanced-learn.readthedocs.io/en/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html (https://imbalanced-learn.readthedocs.io/en/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html)