In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl

from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import tensorflow as tf
from tensorflow import keras
import statsmodels.api as sm

from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import accuracy_score

import numpy as np
from matplotlib.colors import ListedColormap

def game_result(x):
    if x['score1'] > x['score2']:
        result = 'win'
    elif x['score1'] < x['score2']:
        result = 'loss'
    elif x['score1'] == x['score2']:
        result = 'tie'
    return result

def tf_model(model, X):
    y_hat = model.predict(X)
    y_hat = np.array(np.argmax(y_hat, axis=1))
    y_hat.reshape(len(y_hat),1)
    return y_hat

def plot_decision_regions(X, y, classifier, resolution=0.02, tf=False):
   # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    if tf:
        Z = tf_model(classifier, np.array([xx1.ravel(), xx2.ravel()]).T
    else:
        Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.figure(figsize=(20,10))
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot all samples
    for idx, cl in enumerate(np.unique(y)):
        print(f"{cl} is color {colors[idx]} and shape {markers[idx]}")
```

```
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.7,
                    c=cmap(idx),
                    marker=markers[idx],
                    label=cl)
    if tf:
        y_hat= tf_model(classifier, X)
    else:
        y_hat = classifier.predict(X)
    print(accuracy_score(y, y_hat))
```

Objective

This analysis will use the Soccer Power Index (SPI) for the home and away teams to classify whether the game is expected to end in a win, loss or tie for the home team.

Dataset

The data set is the Soccer Power Index data from fivethirtyeight. This link (https://fivethirtyeight.com/methodology/how-our-club-soccer-predictions-work/) contains a detailed write up on the SPI model that fivethirtyeight uses as well here is a link (https://www.espn.com/world-cup/story/_/id/4447078/ce/us/guide-espn-spi-ratings) to a description written by Nate himself. At a high level the SPI gives a prediction on how probability of winning (0 to 100).

The data set contains following columns (I was unabled to find an offical data dictionary so some of these are my interpretation):

- date - Date of the match
- league_id - An id indicating the league in which the game is being played
- league - The name of the league
- team1 - The home team of the game
- team2 - The away team of the game
- spi1 - The SPI for team 1
- spi2 - The SPI for team 2
- prob1 - The predicted probability of a win for team 1
- prob2 - The predicted probability of a win for team 2
- probtie - The predicted probability of a time between the teams
- proj_score1 - The projected number of goals for team 1
- proj_score2 - The projected number of goals for team 2
- importance1 - An adjustmentment factor applied in the SPI calculation for team 1
- importance2 - An adjustmentment factor applied in the SPI calculation for team 2
- score1 - The number of goals for team 1
- score2 - The number of goals for team 2
- xg1 - The expected goals for team 1 based on their play/chances in the game
- xg2 - The expected goals for team 2 based on their play/chances in the game
- nsxg1 - The non-shot expected goals model for team 1 based on their play/chances in the game

- nsxg2 - The non-shot expected goals model for team 2 based on their play/chances in the game
- adj_score1 - The adjusted score for team 1 based on play
- adj_score2 - The adjusted score for team 2 based on play

Getting & Preparing Data

Data was retried from here (https://data.fivethirtyeight.com/). Below describes the steps in data prepartion

1. Download and unzip data into data folder
2. Use Pandas built in csv reader function
3. Use the head and transpose function to see the first five enteries in the data set
4. Calculate the difference in win probabilities between the home and away team, assign it to a column named "prob_diff"
5. Calculate the difference in the goals between the home and away team, assign it to a column named "score_diff"

In [2]:
```
matches = pd.read_csv('data/raw/spi_matches.csv')
matches[matches['league']=='Major League Soccer'].head(5).T
```

Out[2]:

	1412	1416	1429	1431	1436
date	2017-03-03	2017-03-04	2017-03-04	2017-03-04	2017-03-04
league_id	1951	1951	1951	1951	1951
league	Major League Soccer	Major League Soccer	Major League Soccer	Major League Soccer	Major League Soccer
team1	Portland Timbers	Columbus Crew	Los Angeles Galaxy	Real Salt Lake	Colorado Rapids
team2	Minnesota United FC	Chicago Fire	FC Dallas	Toronto FC	New England Revolution
spi1	31.44	30.88	37.45	26.84	28.34
spi2	27.19	24.8	37.23	44.3	31.02
prob1	0.5244	0.5337	0.4962	0.2834	0.4945
prob2	0.2284	0.2338	0.2385	0.4561	0.2302
probtie	0.2472	0.2324	0.2652	0.2605	0.2754
proj_score1	1.79	2.22	1.48	1.28	1.31
proj_score2	1.08	1.34	0.95	1.51	0.81
importance1	20.4	22.2	18.2	17.7	20.9
importance2	21.1	18.9	16.3	13.9	21.8
score1	5	1	1	0	1
score2	1	1	2	0	0
xg1	2.9	1.62	1.59	1.53	0.92
xg2	0.47	0.74	0.41	1.66	0.39
nsxg1	1.59	1.55	1.5	1.2	1.23
nsxg2	1.57	1.01	0.7	0.6	0.69
adj_score1	3.9	1.05	1.05	0	1.05
adj_score2	1.05	1.05	2.1	0	0

In [3]:
```
# Filter the data set to only matches with a result
matches_played = matches[~(matches['score1'].isnull()) & ~(matches['scc
# Use the game result function to convert the score of a game into a ga
matches_played['result'] = matches_played[['score1','score2']].apply(ga
```

```
/Users/matt.stonkus/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

In [4]:
```
# Seperate the dataset into an X and y variable
X = matches_played[['spi1', 'spi2']]
y = matches_played['result']

#Seperate into a test and train set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# Scale the X data set to have a mean 0 and standard deviation of 1
# This improve the learning via gradient descent
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = pd.DataFrame(scaler.transform(X_train), columns=X.colu
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X.column
X_scaled = pd.DataFrame(scaler.transform(X), columns=X.columns)

# Realign the index between the X and y sets
X_train_scaled.index = y_train.index
X_test_scaled.index = y_test.index
```

Data Visualizations

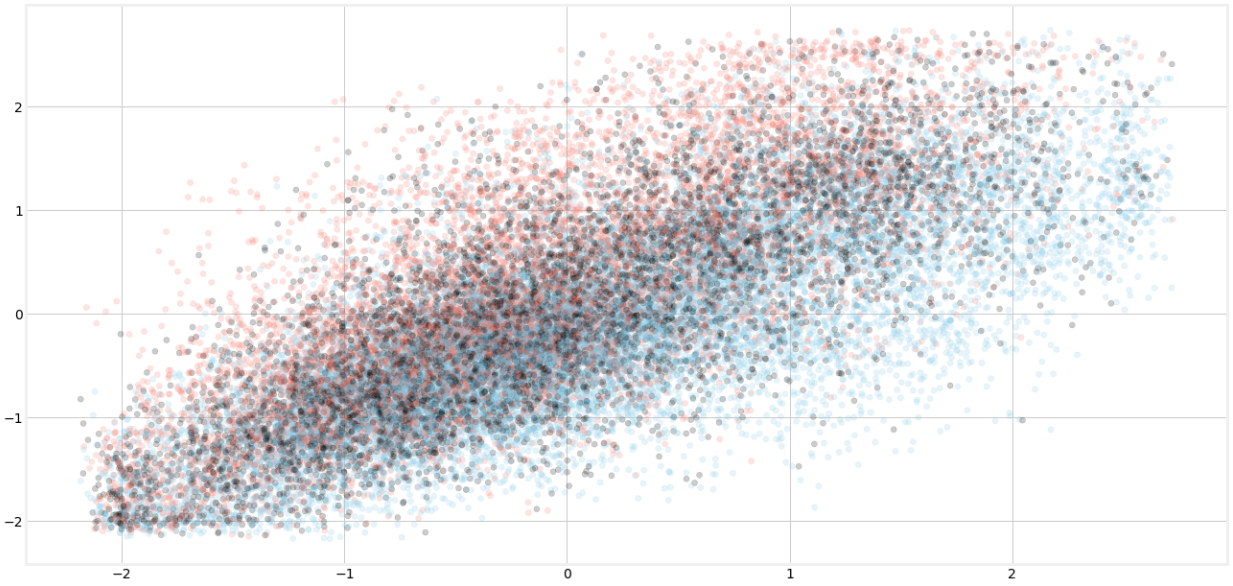
Scatter Plot

An alpha of 0.2 was used in order to better see data distribution/density, darker areas will have more overlapping data points.

Reviewing the graph you can see wins (sky blue) typically are below the 45 degree line (when the home team has a higher SPI than the away team), losses are above the 45 degree line (Away team has a higher SPI than the home team). However close to the 45 degree line, when the teams are more evenly matched it is a mix of wins, losses and ties.

In [5]:
```
color_map = {'win':'skyblue', 'tie':'black', 'loss':'salmon'}
```

In [6]:
```
plt.style.use('fivethirtyeight')
fig, axs = plt.subplots(figsize=(20,10))
axs.scatter(x = X_scaled['spi1'],
            y = X_scaled['spi2'],
            c = y.map(color_map),
            alpha=0.2)
axs.set_facecolor('white')
fig.patch.set_facecolor('white')
plt.show()
```



Modeling

A variety of different classification models are used. For each model its classification boundries are plotted using the training and testing data set. As well the a classification report is generate for each model on test and train data.

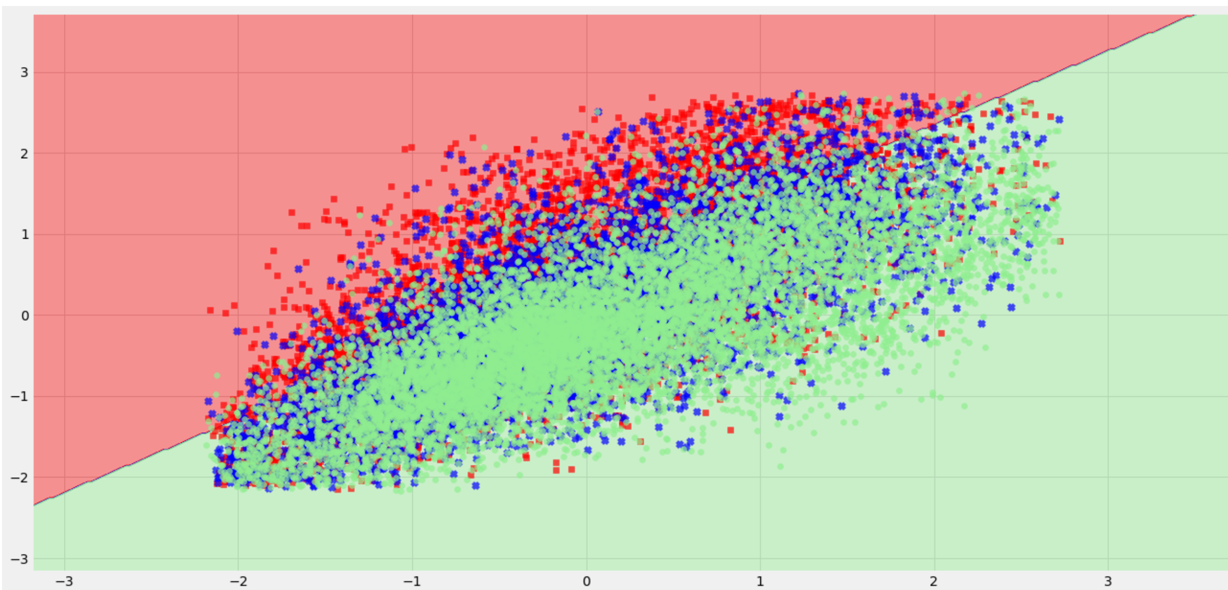
In [7]:
```
from sklearn import svm
result = {'win':3,'tie':1,'loss':0}
y_map = y.map(result)
y_train  = np.array(y_train.map(result))
X_train = np.array(X_train_scaled)
y_test  = np.array(y_test.map(result))
X_test = np.array(X_test_scaled)

C = 1.0  # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=C).fit(X_train, y_train)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X_train, y_train)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X_train, y_train)
logit = LogisticRegression(multi_class='ovr', solver='lbfgs', fit_inter
```

Linear SVC

In [8]:
```
plot_decision_regions(X_train, y_train, svc)
```

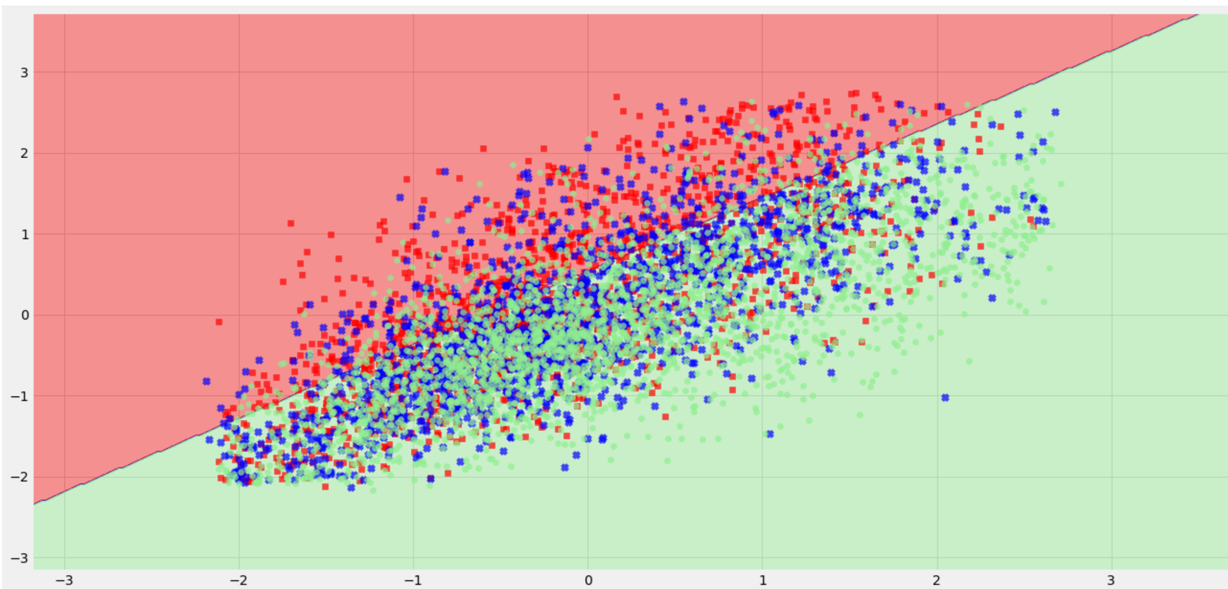
```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
```

```
0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.5082413121143229
```



In [9]:
```
plot_decision_regions(X_test, y_test, svc)
```

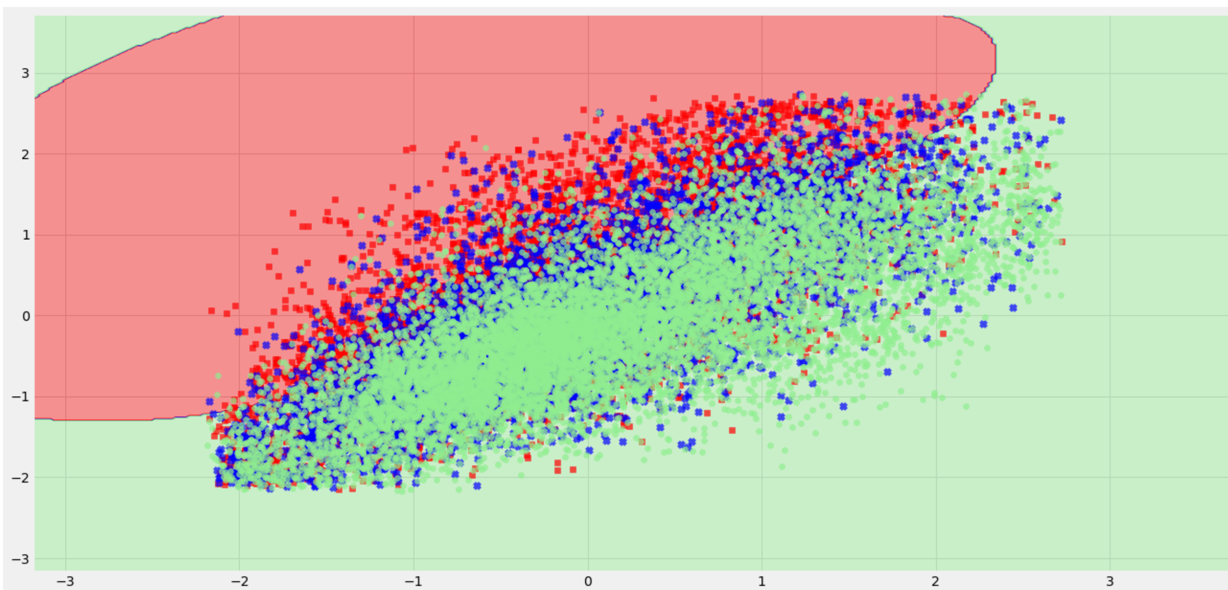
```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.49772653458915234
```



RBF SVC

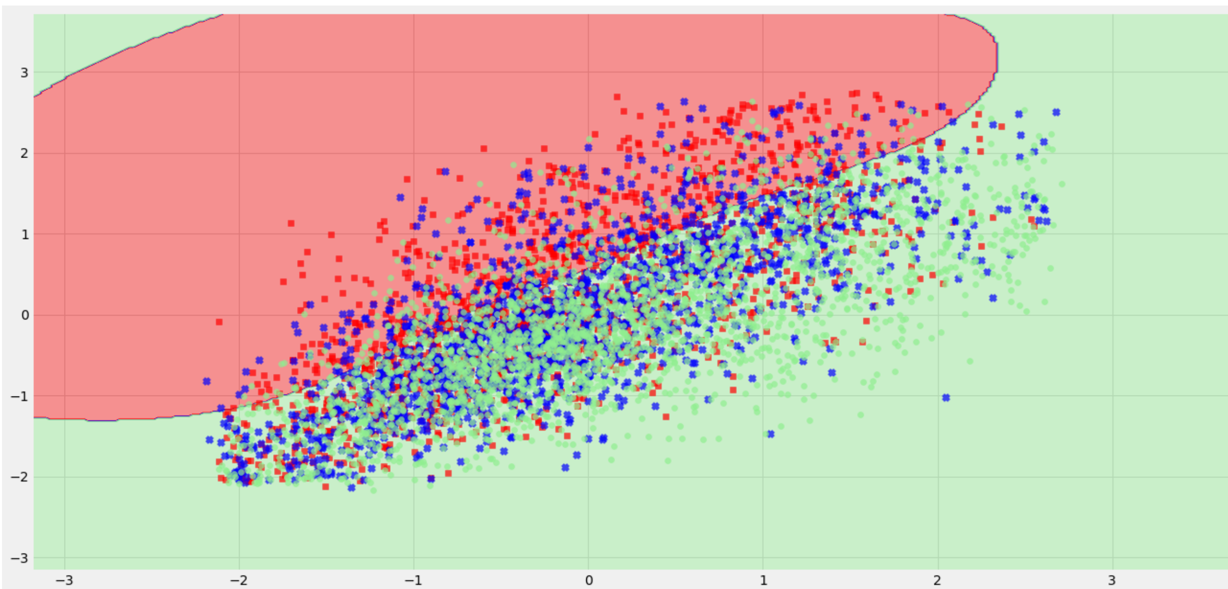
In [10]:
```
1 plot_decision_regions(X_train, y_train, rbf_svc)
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.5078759337447223
```



In [11]:
```
plot_decision_regions(X_test, y_test, rbf_svc)
```

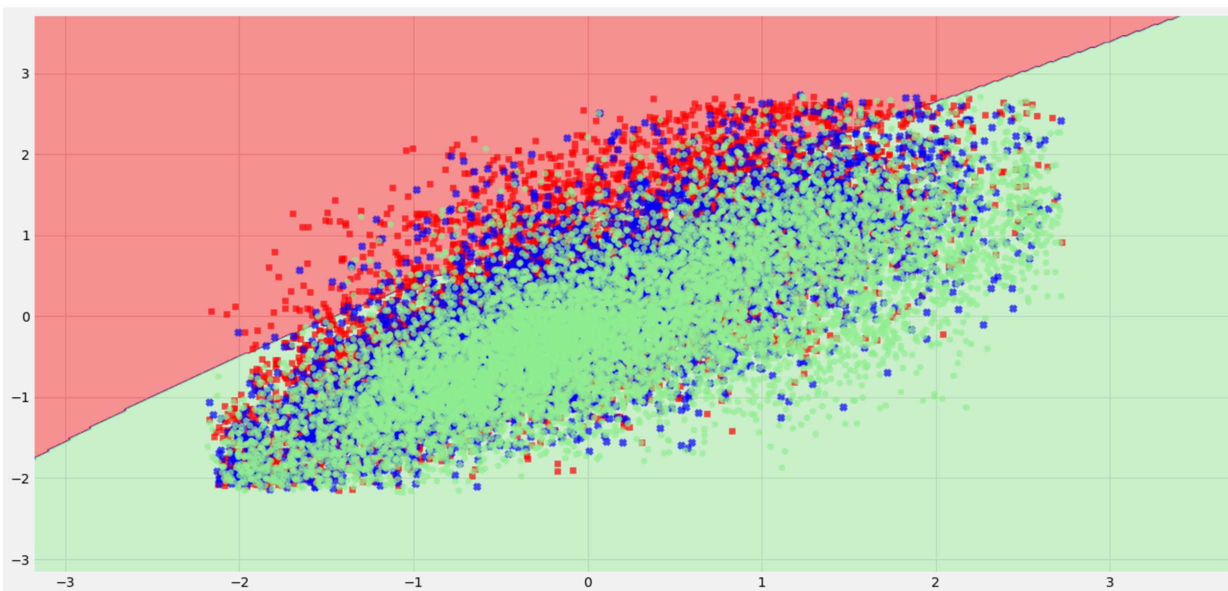
```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.4967521922702176
```



Poly SVC

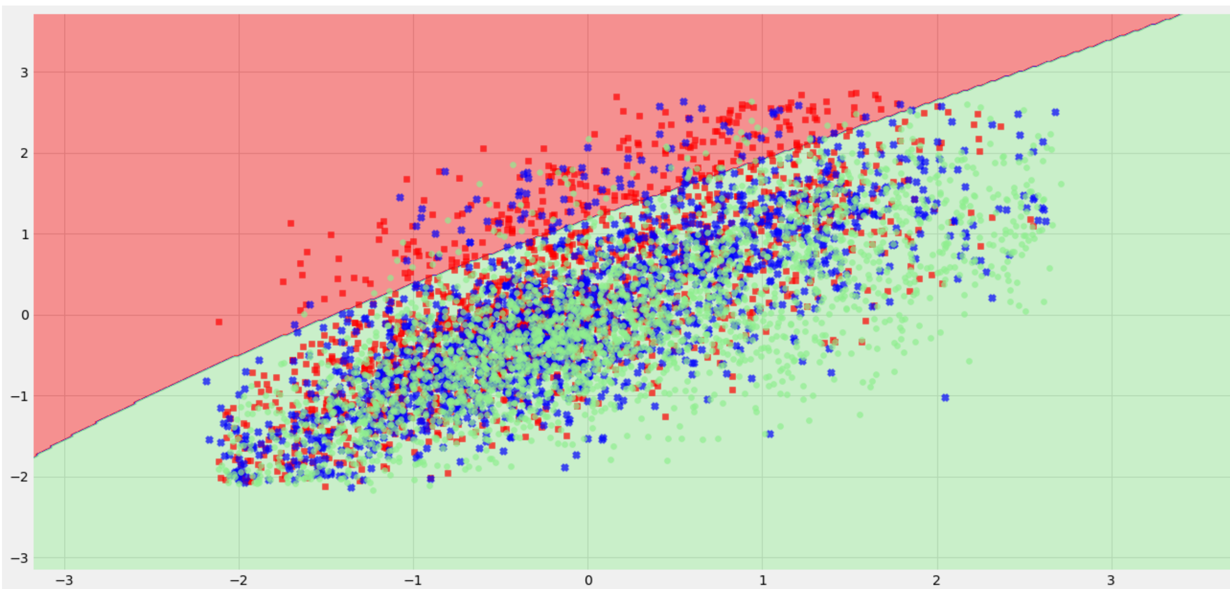
In [12]:
```
plot_decision_regions(X_train, y_train, poly_svc)
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.4851818772328678
```



In [13]:
```
1 plot_decision_regions(X_test, y_test, poly_svc)
```

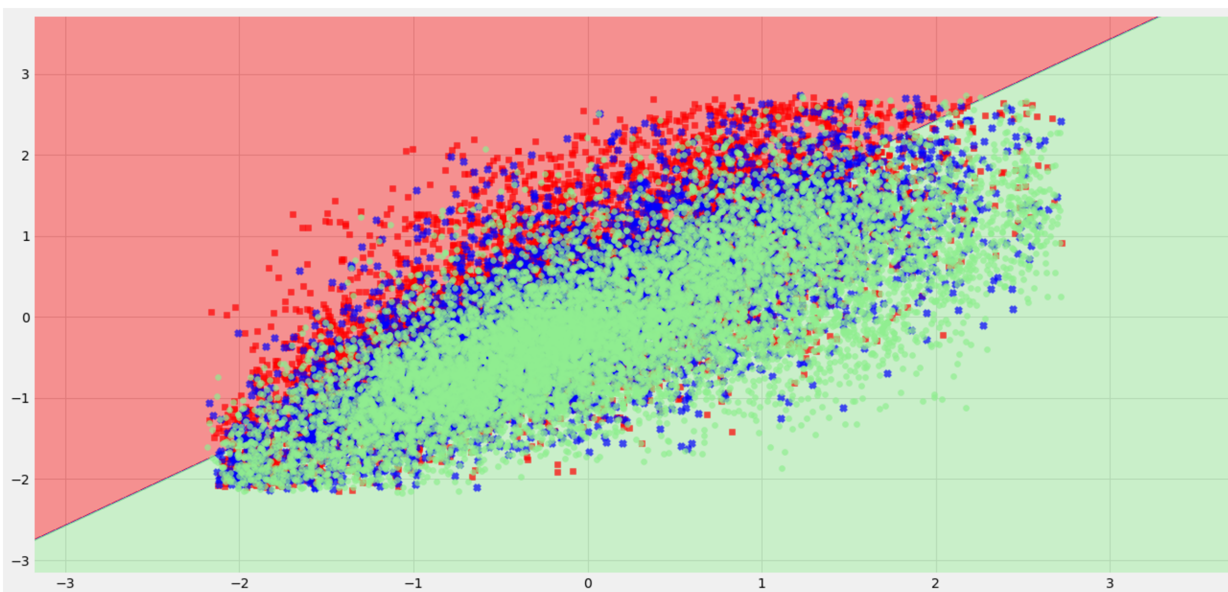
```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.


0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.47596622279961026
```



Logit

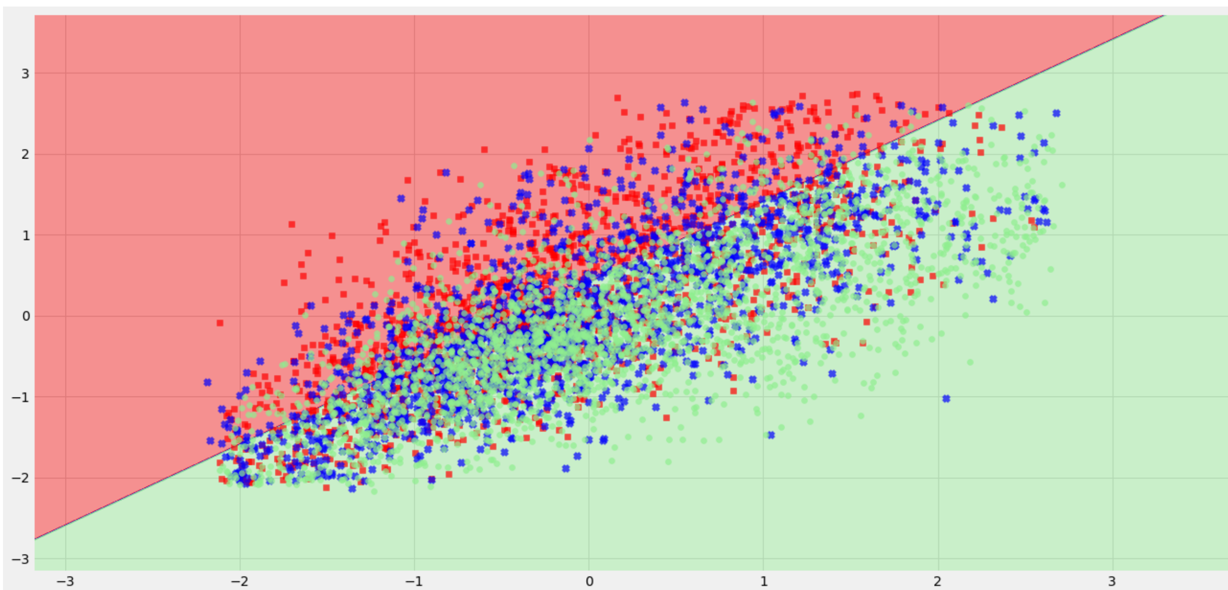
In [14]:
```
plot_decision_regions(X_train, y_train, logit)
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.


0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.5079571289379669
```



In [15]:
```
plot_decision_regions(X_test, y_test, logit)
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.5
```



Tensorflow

In [16]:
```
model = keras.Sequential([
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(10, activation='tanh'),
    keras.layers.Dense(4)
])
```

In [17]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
              metrics=['accuracy'])
```

In [18]:
```
y_train = y_train.astype(int)
model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
770/770 [==============================] - 0s 574us/step - loss: 1.0411 -
accuracy: 0.4997
Epoch 2/10
770/770 [==============================] - 0s 563us/step - loss: 1.0046 -
accuracy: 0.5063
Epoch 3/10
770/770 [==============================] - 0s 561us/step - loss: 1.0033 -
accuracy: 0.5065
Epoch 4/10
770/770 [==============================] - 0s 561us/step - loss: 1.0031 -
accuracy: 0.5067
Epoch 5/10
770/770 [==============================] - 0s 577us/step - loss: 1.0026 -
accuracy: 0.5061
Epoch 6/10
770/770 [==============================] - 0s 560us/step - loss: 1.0020 -
accuracy: 0.5065
Epoch 7/10
770/770 [==============================] - 0s 577us/step - loss: 1.0021 -
accuracy: 0.5058
Epoch 8/10
770/770 [==============================] - 0s 572us/step - loss: 1.0015 -
accuracy: 0.5068
Epoch 9/10
770/770 [==============================] - 0s 570us/step - loss: 1.0017 -
accuracy: 0.5085
Epoch 10/10
770/770 [==============================] - 0s 568us/step - loss: 1.0016 -
accuracy: 0.5063
```

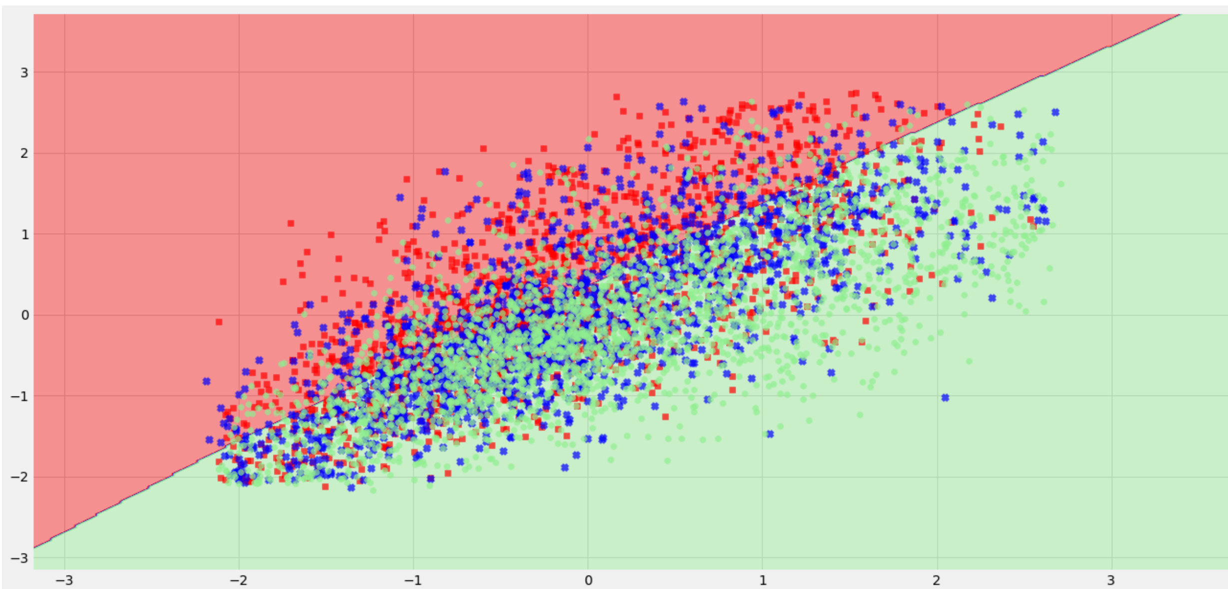
Out[18]:
```
<tensorflow.python.keras.callbacks.History at 0x148583128>
```

In [19]:
```
test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
193/193 - 0s - loss: 1.0123 - accuracy: 0.4990

Test accuracy: 0.4990256428718567
```

In [20]:
```
plot_decision_regions(X_test, y_test, model, tf=True)
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.

0 is color red and shape s
1 is color blue and shape x
3 is color lightgreen and shape o
0.49902565768106527
```



In [ ]:
```

```