Explore Attribute Variables

Select four arbitrary features (any four will do) and get paired plots.

In [ ]:
```
# Creates a grid using Seaborn's PairGrid()
g = sns.PairGrid(
    trn,
    vars=["P25", "P30", "P45", "P60"],
    hue="y",
    diag_sharey=False,
    palette=["red", "green", "blue"])

# Adds histograms on the diagonal
g.map_diag(plt.hist)

# Adds density plots above the diagonal
g.map_upper(sns.kdeplot)

# Adds scatterplots below the diagonal
g.map_lower(sns.scatterplot)

# Adds a legend
g.add_legend()
```

SAVE DATA

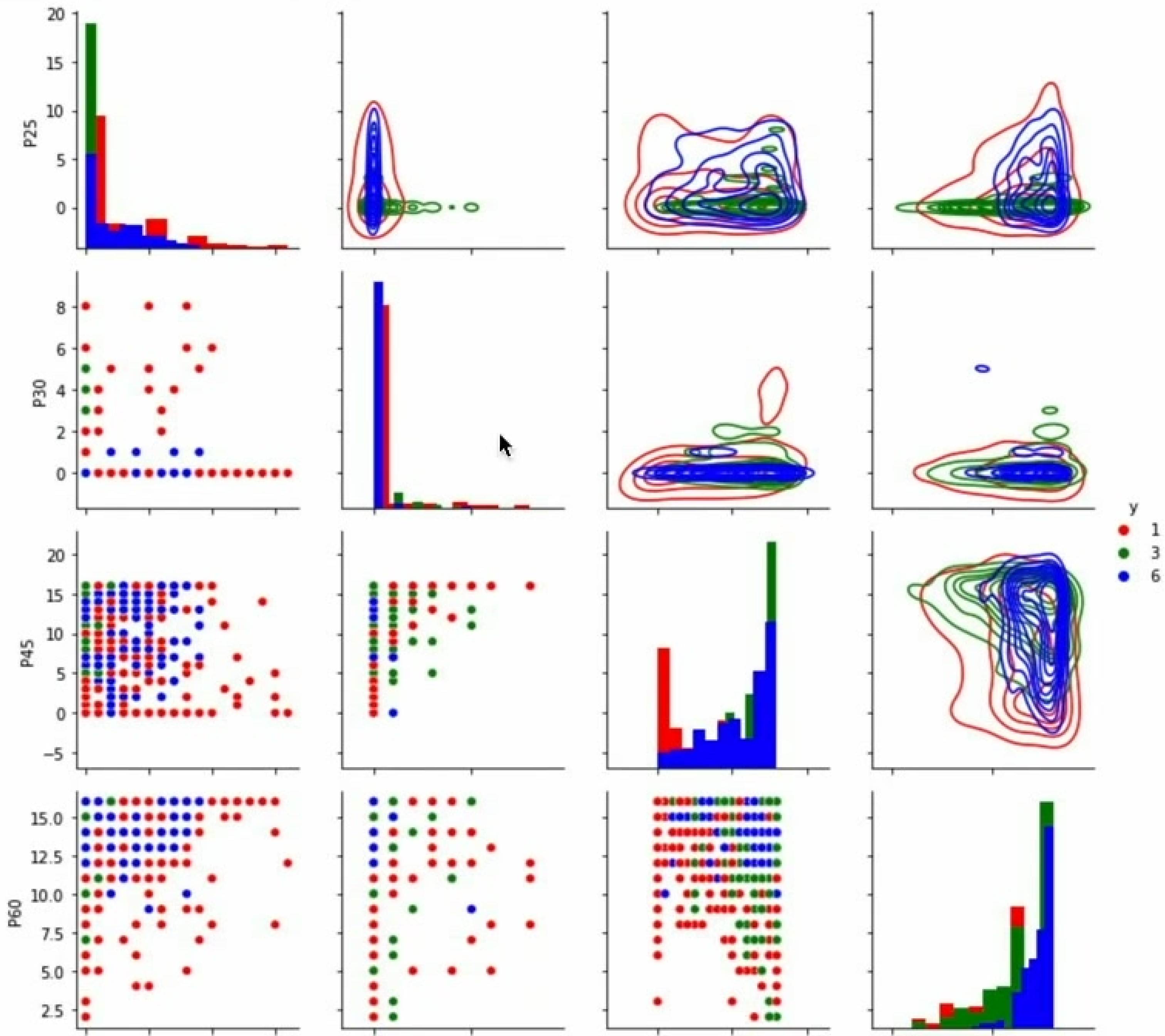
Save `df`, `trn`, and `tst` to CSV files to be used later.

In [ ]:
```
df.to_csv('data/optdigits.csv', sep=',', index=False)
trn.to_csv('data/optdigits_trn.csv', sep=',' ,index=False)
tst.to_csv('data/optdigits_tst.csv', sep=',' ,index=False)
```

CLEAN UP

- If desired, clear the results with Cell > All Output > Clear.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3



In [14]:
```
print('Accuracy on training data = '
    + str("{:.2%}".format(lda.score(X_trn, y_trn))))
```

```
Accuracy on training data = 99.63%
```

TEST MODEL

In this phase, we'll take the LDA model developed above and do the following:

1. Transform the test set using the trained model.
2. Plots the transformed data.
3. Find the prediction accuracy on the testing data.

In [ ]:
```
# Uses the trained model to transform the test data
tst_tf = lda.transform(X_tst)

# Plots the projected data set on the first two discriminant functions and colors by class
sns.scatterplot(
    x=tst_tf[:, 0],
    y=tst_tf[:, 1],
    style=y_tst,
    hue=y_tst,
    palette=['red', 'green', 'blue'])
```

Get the accuracy of the model on the testing data using `score()` and display as percentage with two decimal places.

In [ ]:
```
print('Accuracy on testing data = '
    + str("{:.2%}".format(lda.score(X_tst, y_tst))))
```

CLEAN UP

- If desired, clear the results with Cell > All Output > Clear.
- Save your work by clicking the Save and Checkpoint icon (below File).
- Shut down the Python kernel and close the file by selecting File > Close and Halt.

Data Science Foundations: Data Mining in Python

with **Barton Poulson**

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

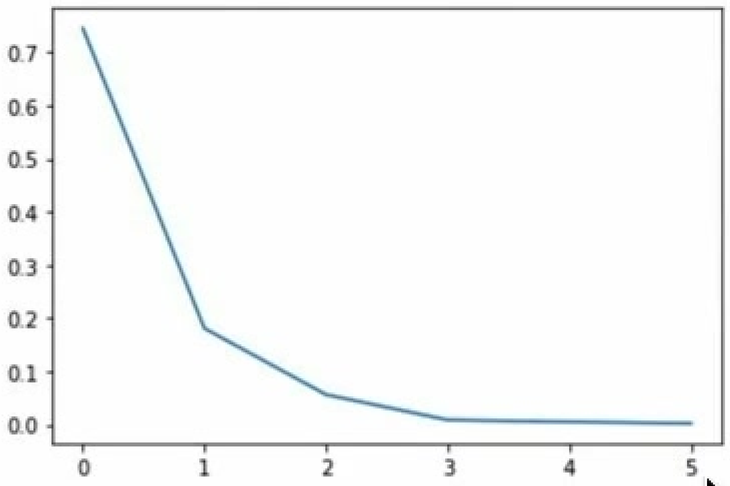
PRINCIPAL COMPONENT ANALYSIS

In [5]:
```
# Sets up the PCA object
pca = PCA()

# Transforms the data ('tf' = 'transformed')
df_tf = pca.fit_transform(df)

# Plot the variance explained by each component
plt.plot(pca.explained_variance_ratio_)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7fec4399ccd0>]
```



In [ ]:
```
# Plots the projected data set on the first two principal components and colors by class
sns.scatterplot(
    x=df_tf[:, 0],
    y=df_tf[:, 1])
```

CLEAN UP

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
grid = GridSearchCV(
    dt,
    {'max_leaf_nodes': param,
     'criterion': ['entropy','gini']})

# Fits the grid to the training data
grid.fit(X_trn,y_trn)

# Stores the optimum model in best_dt
best_dt = grid.best_estimator_

# Displays the optimum model
best_dt.get_params()
```

Out[6]:

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': 38,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'presort': 'deprecated',
 'random_state': 1,
 'splitter': 'best'}
```

Plot Accuracy Against Various Parameters

The code below creates a plot of accuracy against various values of `max_leaf_nodes`. The `gini` and `entropy` measures are plotted separately.

In [ ]:

```
# Plots the mean accuracy against max_leaf_nodes
sns.relplot(
    data=pd.DataFrame.from_dict(grid.cv_results_, orient='columns'),
    kind='line',
    x='param_max_leaf_nodes',
```

Data Science Foundations: Data Mining in Python

with **Barton Poulson**