```python
"""
Statistical Risk Model
Author: Evan Fotopoulos
"""


import numpy as np
import pandas as pd
from sklearn.decomposition import PCA

class Risk_Model():
    """
    Description:
        class to calculate statistical risk model from historical returns.

    Arguments:
        returns {pd.DataFrame} -- returns, stocks as columns, dates as index
        num_factors {int} -- Number of PCs to use

    Keyword Arguments:
        annualisation_factor {int} -- Number of rows per year in returns (default: {252})

    Returns:
        component_returns {pd.DataFrame} --
        component_cov_matrix {} --
        stock_betas {} --

    Functions:
        idiosyncratic_variance_matrix
        idiosyncratic_variance_vector
        predict_portfolio_vol
    """

    def __init__(self, returns, num_factors, annualisation_factor = 252):
        self.returns = returns
        self.num_factors = num_factors
        self.annualisation_factor = annualisation_factor
        self.stocks = returns.columns
        self.dates = returns.index

        self.pca = PCA(n_components = num_factors).fit(returns)

        #Apply dimensionality reduction to returns
        self.component_returns = pd.DataFrame(self.pca.transform(returns),
                                                self.dates,
                                                np.arange(1,num_factors+1))

        self.component_cov_matrix = pd.DataFrame(np.diag(self.component_returns.var(axis = 0,
ddof = 1)
                                                                    ),
                                                np.arange(1,num_factors+1),
                                                np.arange(1,num_factors+1))

        self.stock_betas = pd.DataFrame(self.pca.components_.T,
                                        self.stocks,
                                        np.arange(1,num_factors+1))

    def idiosyncratic_variance_matrix(self, stocks):
        try:
            stock_betas = self.stock_betas.loc[stocks]
            common = pd.DataFrame(np.dot(self.component_returns, stock_betas.T),
                                            self.dates,
                                            stocks)
            residuals = self.returns - common
            return pd.DataFrame(np.diag(np.var(residuals))*self.annualisation_factor,
                                stocks,
                                stocks)
        except KeyError:
            print("stock list contains stock not in returns data used in PCA")
```

```python
                return np.nan


        def idiosyncratic_variance_vector(self, stocks):
            return pd.DataFrame(np.diagonal(self.idiosyncratic_variance_matrix(stocks)),
                                stocks)

        def predict_portfolio_vol(self, weights):
            try:
                npstock_betas = self.stock_betas.loc[(weights.index)].values
                npcommon = npstock_betas.dot(self.component_cov_matrix.values.dot(npstock_betas.T))
                npidio = self.idiosyncratic_variance_matrix(weights.index).values
                npweights = weights.values
                return np.sqrt(npweights.T.dot(npcommon + npidio).dot(npweights))[0][0]
            except KeyError:
                print("portfolio contains stock not in returns data used in PCA")
                return np.nan

            """
            Get the predicted portfolio risk

            Formula for predicted portfolio risk is sqrt(X.T(BFB.T + S)X) where:
            X is the portfolio weights
            B is the factor betas
            F is the factor covariance matrix
            S is the idiosyncratic variance matrix

            Parameters
            ----------
            factor_betas : DataFrame
                Factor betas
            factor_cov_matrix : 2 dimensional Ndarray
                Factor covariance matrix
            idiosyncratic_var_matrix : DataFrame
                Idiosyncratic variance matrix
            weights : DataFrame
                Portfolio weights

            Returns
            -------
            predicted_portfolio_risk : float
                Predicted portfolio risk
            """
```