* import pandas as pd
* import numpy as np
* import matplotlib.pyplot as plt
* from scipy.signal import find\_peaks, savgol\_filter
* from sklearn.cross\_decomposition import PLSRegression
* from sklearn.preprocessing import StandardScaler
* from sklearn.metrics import r2\_score
* import seaborn as sns
* # Load training data
* print("Loading and preprocessing data...")
* train = pd.read\_excel('0611-train-new.xlsx')
* train\_wavenumber = train['wave number']
* # Load prediction data
* pred = pd.read\_excel('0611-verify & predict-new.xlsx')
* pred\_wavenumber = pred['wave number']
* # Get column names
* pred\_abs\_cols = [col for col in pred.columns if col != 'wave number']
* train\_abs\_cols = [col for col in train.columns if col not in ['name', 'c\_K2CO3', 'c\_KHCO3', 'Total', 'wave number']]
* print(f"\nProcessing {len(pred\_abs\_cols)} unknown samples...")
* # Handle missing values
* train = train.fillna(method='ffill').fillna(method='bfill')
* pred = pred.fillna(method='ffill').fillna(method='bfill')
* # Function to smooth spectra and find major peaks
* def process\_spectrum(spectrum, window=51, polyorder=3, prominence=0.5, width=10):
* # Apply Savitzky-Golay filter for smoothing
* spectrum\_smooth = savgol\_filter(spectrum, window, polyorder)
* # Find peaks with more stringent criteria
* peaks, properties = find\_peaks(spectrum\_smooth,
* prominence=prominence,
* width=width,
* height=np.mean(spectrum\_smooth))
* return spectrum\_smooth, peaks, properties
* # Plot all absorption spectra for training file with major peaks
* plt.figure(figsize=(16, 10))
* for col in train\_abs\_cols:
* spectrum = train[col].values
* spectrum\_smooth, peaks, \_ = process\_spectrum(spectrum)
* plt.plot(train\_wavenumber, spectrum\_smooth, label=col, alpha=0.7)
* plt.plot(train\_wavenumber.iloc[peaks], spectrum\_smooth[peaks], 'rx')
* plt.xlabel('Wavenumber')
* plt.ylabel('Absorbance')
* plt.title('Training File: All Absorption Spectra (Smoothed)')
* plt.legend(fontsize=7, ncol=3)
* plt.tight\_layout()
* plt.savefig('train\_all\_spectra\_smooth.png', dpi=300)
* plt.close()
* # Plot peaks for each absorption column in training file
* for col in train\_abs\_cols:
* plt.figure(figsize=(8, 4))
* spectrum = train[col].values
* spectrum\_smooth, peaks, \_ = process\_spectrum(spectrum)
* plt.plot(train\_wavenumber, spectrum\_smooth, label='Smoothed spectrum')
* plt.plot(train\_wavenumber.iloc[peaks], spectrum\_smooth[peaks], 'rx', label='Major Peaks')
* plt.xlabel('Wavenumber')
* plt.ylabel('Absorbance')
* plt.title(f'Training: {col} Major Peaks')
* plt.legend()
* plt.tight\_layout()
* plt.savefig(f'train\_peaks\_{col}.png', dpi=200)
* plt.close()
* # Plot all absorption spectra for prediction file with major peaks
* plt.figure(figsize=(16, 10))
* for col in pred\_abs\_cols:
* spectrum = pred[col].values
* spectrum\_smooth, peaks, \_ = process\_spectrum(spectrum)
* plt.plot(pred\_wavenumber, spectrum\_smooth, label=col, alpha=0.7)
* plt.plot(pred\_wavenumber.iloc[peaks], spectrum\_smooth[peaks], 'rx')
* plt.xlabel('Wavenumber')
* plt.ylabel('Absorbance')
* plt.title('Prediction File: All Absorption Spectra (Smoothed)')
* plt.legend(fontsize=7, ncol=3)
* plt.tight\_layout()
* plt.savefig('predict\_all\_spectra\_smooth.png', dpi=300)
* plt.close()
* # Plot peaks for each absorption column in prediction file
* for col in pred\_abs\_cols:
* plt.figure(figsize=(8, 4))
* spectrum = pred[col].values
* spectrum\_smooth, peaks, \_ = process\_spectrum(spectrum)
* plt.plot(pred\_wavenumber, spectrum\_smooth, label='Smoothed spectrum')
* plt.plot(pred\_wavenumber.iloc[peaks], spectrum\_smooth[peaks], 'rx', label='Major Peaks')
* plt.xlabel('Wavenumber')
* plt.ylabel('Absorbance')
* plt.title(f'Prediction: {col} Major Peaks')
* plt.legend()
* plt.tight\_layout()
* plt.savefig(f'predict\_peaks\_{col}.png', dpi=200)
* plt.close()
* # Prepare training data for PLSR
* # Transpose the data to get samples as rows
* X\_train = train[train\_abs\_cols].T.values
* Y\_train = train[['c\_K2CO3', 'c\_KHCO3']].iloc[:len(train\_abs\_cols)].values
* scaler\_X = StandardScaler()
* scaler\_Y = StandardScaler()
* X\_train\_scaled = scaler\_X.fit\_transform(X\_train)
* Y\_train\_scaled = scaler\_Y.fit\_transform(Y\_train)
* # Find optimal number of components
* max\_components = min(10, min(X\_train.shape))
* mse\_values = []
* for n\_comp in range(1, max\_components + 1):
* pls = PLSRegression(n\_components=n\_comp)
* pls.fit(X\_train\_scaled, Y\_train\_scaled)
* Y\_pred = pls.predict(X\_train\_scaled)
* mse = np.mean((Y\_train\_scaled - Y\_pred) \*\* 2)
* mse\_values.append(mse)
* optimal\_components = np.argmin(mse\_values) + 1
* print(f"\nOptimal number of PLS components: {optimal\_components}")
* # Fit final PLSR model
* pls = PLSRegression(n\_components=optimal\_components)
* pls.fit(X\_train\_scaled, Y\_train\_scaled)
* # Training performance
* Y\_train\_pred\_scaled = pls.predict(X\_train\_scaled)
* Y\_train\_pred = scaler\_Y.inverse\_transform(Y\_train\_pred\_scaled)
* r2\_train\_k2co3 = r2\_score(Y\_train[:, 0], Y\_train\_pred[:, 0])
* r2\_train\_khco3 = r2\_score(Y\_train[:, 1], Y\_train\_pred[:, 1])
* print("\nTraining Set Performance:")
* print(f"R² score for K2CO3: {r2\_train\_k2co3:.3f}")
* print(f"R² score for KHCO3: {r2\_train\_khco3:.3f}")
* # Predict concentrations for each sample
* predictions = []
* for sample in pred\_abs\_cols:
* # Get spectrum for this sample and transpose it
* spectrum = pred[sample].values.reshape(1, -1)
* # Scale the spectrum
* spectrum\_scaled = scaler\_X.transform(spectrum)
* # Predict concentrations
* pred\_conc\_scaled = pls.predict(spectrum\_scaled)
* pred\_conc = scaler\_Y.inverse\_transform(pred\_conc\_scaled)
* # Ensure non-negative concentrations
* pred\_conc = np.maximum(pred\_conc, 0)
* predictions.append({
* 'Sample': sample,
* 'Predicted\_K2CO3': pred\_conc[0, 0],
* 'Predicted\_KHCO3': pred\_conc[0, 1],
* 'Total\_Concentration': pred\_conc[0, 0] + pred\_conc[0, 1]
* })
* # Create results DataFrame
* results\_df = pd.DataFrame(predictions)
* results\_df = results\_df.sort\_values('Total\_Concentration', ascending=False)
* # Print summary statistics
* print("\nPredicted Concentration Summary Statistics:")
* print(results\_df[['Predicted\_K2CO3', 'Predicted\_KHCO3', 'Total\_Concentration']].describe())
* # Save detailed results
* print("\nSaving detailed results to 'predicted\_concentrations\_detailed.xlsx'...")
* results\_df.to\_excel('predicted\_concentrations\_new.xlsx', index=False)
* # Create heatmap
* plt.figure(figsize=(12, 8))
* plt.title('Predicted Concentrations Heatmap')
* sns.heatmap(results\_df[['Predicted\_K2CO3', 'Predicted\_KHCO3']].values.reshape(-1, 2),
* annot=True,
* fmt='.3f',
* xticklabels=['K2CO3', 'KHCO3'],
* yticklabels=results\_df['Sample'],
* cmap='YlOrRd')
* plt.tight\_layout()
* plt.savefig('concentration\_heatmap.png', dpi=300, bbox\_inches='tight')
* plt.close()
* # Plot concentration distribution
* plt.figure(figsize=(10, 6))
* scatter = plt.scatter(results\_df['Predicted\_K2CO3'], results\_df['Predicted\_KHCO3'],
* c=results\_df['Total\_Concentration'], cmap='viridis')
* plt.colorbar(scatter, label='Total Concentration')
* plt.xlabel('Predicted K2CO3 Concentration')
* plt.ylabel('Predicted KHCO3 Concentration')
* plt.title('Distribution of Predicted Concentrations')
* plt.grid(True)
* plt.tight\_layout()
* plt.savefig('concentration\_distribution.png', dpi=300)
* plt.close()
* # Print top 10 highest total concentration samples
* print("\nTop 10 Samples by Total Concentration:")
* print(results\_df.head(10).to\_string(index=False))
* # Save comprehensive report
* print("\nSaving comprehensive report to 'concentration\_prediction\_report.txt'...")
* with open('concentration\_prediction\_report.txt', 'w') as f:
* f.write("Concentration Prediction Report\n")
* f.write("============================\n\n")
* f.write(f"Model Parameters:\n")
* f.write(f"- Number of PLS components: {optimal\_components}\n")
* f.write(f"- Training R² (K2CO3): {r2\_train\_k2co3:.3f}\n")
* f.write(f"- Training R² (KHCO3): {r2\_train\_khco3:.3f}\n\n")
* f.write("Summary Statistics:\n")
* f.write(results\_df.describe().to\_string())
* f.write("\n\nDetailed Predictions:\n")
* f.write(results\_df.to\_string())
* print("\nAnalysis complete! Check the following files for results:")
* print("1. predicted\_concentrations\_detailed.xlsx - Detailed predictions for all samples")
* print("2. concentration\_heatmap.png - Visual representation of predicted concentrations")
* print("3. concentration\_distribution.png - Distribution of predicted concentrations")
* print("4. concentration\_prediction\_report.txt - Comprehensive report with all details")