[Data Preprocessing] (CheatSheet)

1. Handling Missing Values

- **Identify Missing Values**: `df.isnull().sum()`
- **Drop Rows with Missing Values**: `df.dropna()`
- **Fill Missing Values with a Specific Value**: `df.fillna(value)`
- **Fill Missing Values with Mean/Median/Mode**: `df.fillna(df.mean())`
- **Interpolate Missing Values**: `df.interpolate()`
- **Forward Fill or Backward Fill**: `df.ffill()` or `df.bfill()`

2. Data Transformation

- **Standardization (Z-Score Normalization)**: `(df - df.mean()) / df.std()`
- **Min-Max Normalization**: `(df - df.min()) / (df.max() - df.min())`
- **Log Transformation**: `np.log(df)`
- **Square Root Transformation**: `np.sqrt(df)`
- **Power Transformation (e.g., Box-Cox)**: `scipy.stats.boxcox(df)`

3. Feature Encoding

- **One-Hot Encoding**: `pd.get_dummies(df)`
- **Label Encoding**: `sklearn.preprocessing.LabelEncoder()`
- **Binary Encoding**: `category_encoders.BinaryEncoder()`
- **Frequency Encoding**: `df.groupby('column').size() / len(df)`
- **Mean Encoding**: `df.groupby('category')['target'].mean()`

4. Handling Categorical Data

- **Convert to Category Type**: `df['column'].astype('category')`
- **Ordinal Encoding**: `df['column'].cat.codes`
- **Using Pandas' Cut for Binning**: `pd.cut(df['column'], bins)`
- **Using Pandas' QCut for Quantile Binning**: `pd.qcut(df['column'], q)`

By: Waleed Mousa

5. Feature Scaling

- **Robust Scaler**: `sklearn.preprocessing.RobustScaler()`
- **MaxAbsScaler**: `sklearn.preprocessing.MaxAbsScaler()`
- **Normalizer**: `sklearn.preprocessing.Normalizer()`

6. Feature Selection

- **Variance Threshold**: `sklearn.feature_selection.VarianceThreshold()`
- **SelectKBest**: `sklearn.feature_selection.SelectKBest()`
- **Recursive Feature Elimination**: `sklearn.feature_selection.RFE()`
- **SelectFromModel**: `sklearn.feature_selection.SelectFromModel()`
- **Correlation Matrix with Heatmap**: `sns.heatmap(df.corr(), annot=True)`

7. Handling Outliers

- **IQR Method**: `Q1 = df.quantile(0.25); Q3 = df.quantile(0.75); IQR = Q3 - Q1; df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR)))]`
- **Z-Score Method**: `(abs(df - df.mean()) / df.std()) < 3`
- **Winsorizing**: `scipy.stats.mstats.winsorize()`

8. Text Preprocessing (NLP)

- **Tokenization**: `nltk.word_tokenize(text)`
- **Removing Stop Words**: `nltk.corpus.stopwords.words('english')`
- **Stemming**: `nltk.stem.PorterStemmer()`
- **Lemmatization**: `nltk.stem.WordNetLemmatizer()`
- **TF-IDF Vectorization**: `sklearn.feature_extraction.text.TfidfVectorizer()`

9. Time Series Data

- **DateTime Conversion**: `pd.to_datetime(df['column'])`
- **Set DateTime as Index**: `df.set_index('datetime_column')`
- **Resampling for Time Series Aggregation**: `df.resample('D').mean()`
- **Time Series Decomposition**: `statsmodels.tsa.seasonal.seasonal_decompose(df['column'])`

By: Waleed Mousa

10. Data Splitting

- **Train-Test Split**: `sklearn.model_selection.train_test_split()`
- **K-Fold Cross-Validation**: `sklearn.model_selection.KFold()`
- **Stratified Sampling**: `sklearn.model_selection.StratifiedKFold()`

11. Data Cleaning

- **Trimming Whitespace**: `df['column'].str.strip()`
- **Replacing Values**: `df.replace(old_value, new_value)`
- **Dropping Columns**: `df.drop(columns=['column_to_drop'])`
- **Renaming Columns**: `df.rename(columns={'old_name': 'new_name'})`
- **Converting Data Types**: `df.astype({'column': 'new_type'})`

12. Image Data Preprocessing

- **Resizing Images**: `cv2.resize()`
- **Normalizing Pixel Values**: `image / 255.0`
- **Image Augmentation**: `ImageDataGenerator()` in `keras.preprocessing.image`
- **Grayscale Conversion**: `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`

13. Dimensionality Reduction

- **Principal Component Analysis (PCA)**: `sklearn.decomposition.PCA()`
- **t-SNE**: `sklearn.manifold.TSNE()`
- **LDA**: `sklearn.discriminant_analysis.LinearDiscriminantAnalysis()`

14. Dealing with Imbalanced Data

- **Random Over-Sampling**: `imblearn.over_sampling.RandomOverSampler()`
- **Random Under-Sampling**: `imblearn.under_sampling.RandomUnderSampler()`
- **SMOTE**: `imblearn.over_sampling.SMOTE()`

15. Combining Features

- **Polynomial Features**: `sklearn.preprocessing.PolynomialFeatures()`

By: Waleed Mousa

- **Concatenating Features**: `np.concatenate([feature1, feature2], axis=1)`

16. Handling Multivariate Data

- **Granger Causality Test**: `statsmodels.tsa.stattools.grangercausalitytests()`
- **Vector AutoRegression (VAR)**: `statsmodels.tsa.api.VAR()`

17. Signal Processing

- **Fourier Transform**: `np.fft.fft()`
- **Wavelet Transform**: `pywt.Wavelet()`

18. Error Metrics

- **Mean Squared Error (MSE)**: `sklearn.metrics.mean_squared_error()`
- **Mean Absolute Error (MAE)**: `sklearn.metrics.mean_absolute_error()`
- **R-Squared**: `sklearn.metrics.r2_score()`

19. Data Wrangling

- **Pivot Tables**: `df.pivot_table()`
- **Stacking and Unstacking**: `df.stack()`, `df.unstack()`
- **Melting Data**: `pd.melt(df)`

20. Advanced DataFrame Operations

- **Apply Functions**: `df.apply(lambda x: ...)`
- **GroupBy Operations**: `df.groupby('column').aggregate(function)`
- **Merge and Join DataFrames**: `pd.merge(df1, df2, on='key')`, `df1.join(df2, on='key')`

21. Sequence Data Processing

- **Padding Sequences**: `keras.preprocessing.sequence.pad_sequences()`
- **One-Hot Encoding for Sequences**: `keras.utils.to_categorical()`

By: Waleed Mousa

22. Data Verification

- **Assert Statements**: `pd.util.testing.assert_frame_equal()`
- **Data Consistency Check**: `pd.util.testing.assert_series_equal()`

23. Data Aggregation

- **Cumulative Sum**: `df.cumsum()`
- **Cumulative Product**: `df.cumprod()`
- **Weighted Average**: `np.average(values, weights=weights)`

24. Geospatial Data

- **Coordinate Transformation**: `geopandas.GeoDataFrame()`
- **Spatial Join**: `geopandas.sjoin()`
- **Distance Calculation**: `geopy.distance.distance(coord1, coord2)`

25. Handling JSON Data

- **Normalize JSON**: `pd.json_normalize(json_data)`
- **Read JSON**: `pd.read_json('file.json')`
- **To JSON**: `df.to_json()`

26. Handling XML Data

- **Parse XML**: `xml.etree.ElementTree.parse('file.xml')`
- **Find Elements in XML**: `tree.findall('path')`

27. Probability Distributions

- **Normal Distribution**: `np.random.normal()`
- **Uniform Distribution**: `np.random.uniform()`
- **Binomial Distribution**: `np.random.binomial()`

28. Hypothesis Testing

- **t-Test**: `scipy.stats.ttest_ind()`
- **ANOVA Test**: `scipy.stats.f_oneway()`

By: Waleed Mousa

- **Chi-Squared Test**: `scipy.stats.chi2_contingency()`

29. Database Interaction

- **Read SQL Query**: `pd.read_sql_query('SELECT * FROM table', connection)`
- **Write to SQL**: `df.to_sql('table', connection)`

30. Data Profiling

- **Descriptive Statistics**: `df.describe()`
- **Correlation Analysis**: `df.corr()`
- **Unique Value Counts**: `df['column'].value_counts()`
- **Pandas Profiling for Comprehensive Reports**: `pandas_profiling.ProfileReport(df)`

31. Advanced Handling of Missing Values

- **KNN Imputation**: `from sklearn.impute import KNNImputer; imputer = KNNImputer(n_neighbors=5); df_imputed = imputer.fit_transform(df)`
- **Iterative Imputation**: `from sklearn.experimental import enable_iterative_imputer; from sklearn.impute import IterativeImputer; imputer = IterativeImputer(); df_imputed = imputer.fit_transform(df)`

32. Feature Engineering

- **Lag Features for Time Series**: `df['lag_feature'] = df['feature'].shift(1)`
- **Rolling Window Features**: `df['rolling_mean'] = df['feature'].rolling(window=5).mean()`
- **Expanding Window Features**: `df['expanding_mean'] = df['feature'].expanding().mean()`
- **Datetime Features Extraction**: `df['hour'] = df['datetime'].dt.hour`
- **Binning Numeric Features**: `pd.cut(df['numeric_feature'], bins=3, labels=False)`

33. Data Normalization for Text

By: Waleed Mousa

- **Removing Punctuation**: `df['text'].str.replace('[^\w\s]', '', regex=True)`
- **Removing Numbers**: `df['text'].str.replace('\d+', '', regex=True)`
- **Converting to Lowercase**: `df['text'].str.lower()`
- **Removing Whitespaces**: `df['text'].str.strip()`

34. Advanced Text Preprocessing

- **Removing HTML Tags**: `df['text'].str.replace('<.*?>', '', regex=True)`
- **Removing URLs**: `df['text'].str.replace('http\S+|www.\S+', '', regex=True)`
- **Using NLTK for Tokenization**: `nltk.word_tokenize(df['text'])`
- **Using Spacy for Lemmatization**: `spacy.load('en_core_web_sm').lemmatizer(df['text'])`

35. Advanced Feature Scaling

- **Quantile Transformer**: `sklearn.preprocessing.QuantileTransformer()`
- **Power Transformer**: `sklearn.preprocessing.PowerTransformer(method='yeo-johnson')`

36. Balancing Data

- **Oversampling with SMOTE-NC for Categorical Features**: `imblearn.over_sampling.SMOTENC(categorical_features=[0, 2, 3])`
- **Cluster-Based Oversampling**: `imblearn.over_sampling.ClusterCentroids()`

37. Feature Selection Based on Model

- **L1 Regularization for Feature Selection**: `sklearn.linear_model.LogisticRegression(penalty='l1')`
- **Tree-Based Feature Selection**: `sklearn.ensemble.ExtraTreesClassifier()`

38. Data Discretization

- **Discretization into Quantiles**: `pd.qcut(df['feature'], q=4)`

- **K-Means Discretization**: `sklearn.preprocessing.KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='kmeans')`

39. Dealing with Date and Time

- **Time Delta Calculation**: `(df['date_end'] - df['date_start']).dt.days`
- **Extracting Day of Week**: `df['date'].dt.dayofweek`
- **Setting Frequency in Time Series**: `df.asfreq('D')`

40. Handling Geospatial Data

- **Creating Geospatial Features**: `geopandas.GeoDataFrame(df, geometry=geopandas.points_from_xy(df.longitude, df.latitude))`
- **Calculating Distance Between Points**: `df['geometry'].distance(other_point)`

41. Advanced NLP Techniques

- **Named Entity Recognition (NER) with Spacy**: `spacy.load('en_core_web_sm').entity(df['text'])`
- **Topic Modeling with Latent Dirichlet Allocation (LDA)**: `gensim.models.LdaMulticore(corpus, num_topics=10)`

42. Data Decomposition

- **Singular Value Decomposition (SVD)**: `scipy.linalg.svd(matrix)`
- **Non-Negative Matrix Factorization (NMF)**: `sklearn.decomposition.NMF(n_components=2)`

43. Advanced Image Preprocessing

- **Edge Detection in Images (Canny)**: `cv2.Canny(image, threshold1, threshold2)`
- **Image Thresholding**: `cv2.threshold(image, threshold, max_value, cv2.THRESH_BINARY)`

44. Handling JSON and Complex Data Types

By: Waleed Mousa

- **Flattening JSON Nested Structures**: `pd.json_normalize(data, sep='_')`
- **Parsing JSON Strings in DataFrame**: `df['json_col'].apply(lambda x: json.loads(x))`

45. Working with Time Series and Sequences

- **Differencing a Time Series**: `df['value'].diff(periods=1)`
- **Creating Cumulative Features**: `df['cumulative_sum'] = df['value'].cumsum()`

46. Data Validation

- **Asserting Dataframe Equality**: `pd.testing.assert_frame_equal(df1, df2)`
- **Checking DataFrame Schema with Pandera**: `pandera.SchemaModel.validate(df)`

47. Custom Transformations

- **Applying Custom Functions**: `df.apply(lambda row: custom_function(row), axis=1)`
- **Vectorized String Operations**: `df['text'].str.cat(sep=' ')`

48. Feature Extraction from Time Series

- **Fourier Transform for Periodicity**: `np.fft.fft(df['time_series'])`
- **Autocorrelation Features**: `pd.plotting.autocorrelation_plot(df['time_series'])`

49. Working with APIs and Remote Data

- **Reading Data from a REST API**: `pd.read_json(api_endpoint)`
- **Loading Data from Cloud Services (e.g., AWS S3)**: `pd.read_csv('s3://bucket_name/file.csv')`

50. Advanced Data Aggregation

- **Weighted Moving Average**: `df['value'].rolling(window=5).apply(lambda x: np.average(x, weights=[0.1, 0.2, 0.3, 0.2, 0.2]))`
- **Cumulative Maximum or Minimum**: `df['cumulative_max'] = df['value'].cummax()`
- **GroupBy with Custom Aggregation Functions**: `df.groupby('group').agg({'value': ['mean', 'std', custom_agg_function]})`
- **Pivot Table with Multiple Aggregates**: `df.pivot_table(index='group', values='value', aggfunc=['mean', 'sum', 'count'])`