In [21]:

```python
import os
import pandas as pd
import json
import imgkit
import io
import base64
import matplotlib.pyplot as plt
```

In [47]:

```python
# Function to create a small plot and return it as a base64-encoded image
def create_plot(data, title):
    fig, ax = plt.subplots()
    ax.plot(['k2', 'k3', 'k4', 'k5'], data, marker='*', color='red', linewidth=10)
    ax.set_title(title)
    ax.set_ylim(0, 1)
    buf = io.BytesIO()
    plt.savefig(buf, format='png')
    plt.close(fig)
    buf.seek(0)
    img_base64 = base64.b64encode(buf.read()).decode('utf-8')
    return f"<img src='{img_base64}' width='100' height='50'>"
```

In [59]:

```python
def generate_metrics_table(metrics_folder, output_image_path, split=False):
    rows = []
    files = os.listdir(metrics_folder)
    metric_files = [file for file in files if file.startswith("binary_classifier_") and file.endswi

    k_metrics = [None, None, None, None]

    # Read metrics from each file and store in k_metrics list
    for file in metric_files:
        k_value = int(file.split("_")[2].split(".")[0])
        with open(os.path.join(metrics_folder, file), 'r') as f:
            metrics = json.load(f)
        k_metrics[k_value-2] = metrics

    # Iterate through gene families and metrics for each k value
    for gene_family, metrics_data_k2 in k_metrics[0].items():
        metrics_data_k3 = k_metrics[1][gene_family]
        metrics_data_k4 = k_metrics[2][gene_family]
        metrics_data_k5 = k_metrics[3][gene_family]

        # Find the max values for each metric
        max_accuracy = max(metrics_data_k2['accuracy'], metrics_data_k3['accuracy'], metrics_data_k
        max_precision = max(metrics_data_k2['precision'], metrics_data_k3['precision'], metrics_dat
        max_recall = max(metrics_data_k2['recall'], metrics_data_k3['recall'], metrics_data_k4['rec
        max_f1 = max(metrics_data_k2['f1'], metrics_data_k3['f1'], metrics_data_k4['f1'], metrics_c

        # Construct HTML row for each gene family
        row = f"<tr><td><b>{gene_family}</b></td>"

        # Accuracy metrics
        row += f"<td class='lc'>{metrics_data_k2['accuracy']:.3f}</td>" if metrics_data_k2['accurac
        row += f"<td>{metrics_data_k3['accuracy']:.3f}</td>" if metrics_data_k3['accuracy'] != max_
        row += f"<td>{metrics_data_k4['accuracy']:.3f}</td>" if metrics_data_k4['accuracy'] != max_
        row += f"<td>{metrics_data_k5['accuracy']:.3f}</td>" if metrics_data_k5['accuracy'] != max_
        row += f"<td>{create_plot([metrics_data_k2['accuracy'], metrics_data_k3['accuracy'], metric

        # Precision metrics
        row += f"<td class='lc'>{metrics_data_k2['precision']:.3f}</td>" if metrics_data_k2['precis
        row += f"<td>{metrics_data_k3['precision']:.3f}</td>" if metrics_data_k3['precision'] != ma
        row += f"<td>{metrics_data_k4['precision']:.3f}</td>" if metrics_data_k4['precision'] != ma
        row += f"<td>{metrics_data_k5['precision']:.3f}</td>" if metrics_data_k5['precision'] != ma
        row += f"<td>{create_plot([metrics_data_k2['precision'], metrics_data_k3['precision'], metr

        # Recall metrics
        row += f"<td class='lc'>{metrics_data_k2['recall']:.3f}</td>" if metrics_data_k2['recall']
        row += f"<td>{metrics_data_k3['recall']:.3f}</td>" if metrics_data_k3['recall'] != max_reca
        row += f"<td>{metrics_data_k4['recall']:.3f}</td>" if metrics_data_k4['recall'] != max_reca
        row += f"<td>{metrics_data_k5['recall']:.3f}</td>" if metrics_data_k5['recall'] != max_reca
        row += f"<td>{create_plot([metrics_data_k2['recall'], metrics_data_k3['recall'], metrics_da

        # F1 metrics
        row += f"<td class='lc'>{metrics_data_k2['f1']:.3f}</td>" if metrics_data_k2['f1'] != max_f
        row += f"<td>{metrics_data_k3['f1']:.3f}</td>" if metrics_data_k3['f1'] != max_f1 else f"<t
        row += f"<td>{metrics_data_k4['f1']:.3f}</td>" if metrics_data_k4['f1'] != max_f1 else f"<t
        row += f"<td>{metrics_data_k5['f1']:.3f}</td>" if metrics_data_k5['f1'] != max_f1 else f"<t
        row += f"<td>{create_plot([metrics_data_k2['f1'], metrics_data_k3['f1'], metrics_data_k4['f

        row += "</tr>"
        rows.append(row)

    # Function to construct HTML for table
    def construct_html_table(rows):
        return f"""
        <html>
        <head>
            <style>
                table {{
                    width: 100%;
                    border-collapse: collapse;
                    border: 1px solid black;
                }}
                th, td {{
                    border: 1px solid black;
                    padding: 8px;
                    text-align: center;
                    font-size: 14px;
                }}
                th {{
                    background-color: #f2f2f2;
                }}
                .main-col {{
                    text-align: center;
                    background-color: #e6f7ff; /* Soft background color for main columns */
                }}
                .sub-col {{
                    background-color: #f2f2f2; /* Light background color for sub-columns */
                }}
                thead, tbody {{
                    border: 1px solid;
                }}
                .lc {{
                    border-left: 1px solid !important;
                }}
                .metric_style{{
                    text-align: center !important;
                    border: 1px solid !important;
                }}
            </style>
        </head>
        <body>
        <!--<h2>Metrics Table</h2>-->
        <table>
        <thead>
            <tr>
                <th rowspan="2" class="main-col-">Gene (Family)</th>
                <th colspan="5" class="metric_style">Accuracy</th>
                <th colspan="5" class="main-col- metric_style">Precision</th>
                <th colspan="5" class="main-col- metric_style">Recall</th>
                <th colspan="5" class="main-col- metric_style">F1</th>
            </tr>
            <tr>
                <th class="sub-col lc">k=2</th>
                <th class="sub-col">k=3</th>
                <th class="sub-col">k=4</th>
                <th class="sub-col">k=5</th>
                <th class="sub-col">trend</th>
                <th class="sub-col lc">k=2</th>
                <th class="sub-col">k=3</th>
                <th class="sub-col">k=4</th>
                <th class="sub-col">k=5</th>
                <th class="sub-col">trend</th>
                <th class="sub-col lc">k=2</th>
                <th class="sub-col">k=3</th>
                <th class="sub-col">k=4</th>
                <th class="sub-col">k=5</th>
                <th class="sub-col">trend</th>
                <th class="sub-col lc">k=2</th>
                <th class="sub-col">k=3</th>
                <th class="sub-col">k=4</th>
                <th class="sub-col">k=5</th>
                <th class="sub-col">trend</th>
            </tr>
        </thead>
        <tbody>
        {''.join(rows)}
        </tbody>
        </table>
        </body>
        </html>
        """

    # Save HTML table to PNG
    if split:
        # Split rows into two parts
        mid_index = len(rows) // 2
        rows_part1 = rows[:mid_index]
        rows_part2 = rows[mid_index:]

        # Generate HTML content for both parts
        table_html_part1 = construct_html_table(rows_part1)
        table_html_part2 = construct_html_table(rows_part2)

        # Save both parts as separate PNG images
        imgkit.from_string(table_html_part1, output_image_path.replace(".png", "_part1.png"))
        imgkit.from_string(table_html_part2, output_image_path.replace(".png", "_part2.png"))

        return table_html_part1, table_html_part2
    else:
        # Generate HTML content for the whole table
        table_html = construct_html_table(rows)

        # Save as single PNG image
        imgkit.from_string(table_html, output_image_path)

        return table_html
```

In [60]:

```python
# Example usage:
metrics_folder = "./metrics"
output_image_path = "metrics_table_anova.png"
metrics_table = generate_metrics_table(metrics_folder, output_image_path, split=True)
```

```
Loading page (1/2)
Rendering (2/2)
Done
Loading page (1/2)
Rendering (2/2)
Done
```

In [61]:

```python
# Display the HTML table
from IPython.display import display, HTML
display(HTML(metrics_table[0]))
display(HTML(metrics_table[1]))
```

Gene (Family)	Accuracy					Precision					Recall					F1				
	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend
AP2	0.985	0.984	**0.993**	0.987		0.484	0.471	**0.658**	0.512		0.967	0.983	**0.984**	0.950		0.645	0.637	**0.789**	0.665	
ARF	0.988	1.000	1.000	**1.000**		0.540	0.995	0.998	**1.000**		**1.000**	0.996	0.998	0.997		0.701	0.995	0.998	**0.998**	
ARR-B	0.997	**0.999**	0.995	0.993		0.713	**0.896**	0.595	0.526		0.964	0.975	**0.985**	0.983		0.819	**0.934**	0.742	0.685	
B3	0.994	0.995	**0.996**	0.993		0.849	**0.933**	0.917	0.931		**0.983**	0.919	0.969	0.865		0.911	0.926	**0.942**	0.897	
BBR-BPC	0.993	1.000	**1.000**	1.000		0.364	**1.000**	0.996	**1.000**		0.988	0.992	**1.000**	0.984		0.531	0.996	**0.998**	0.992	
BES1	0.970	**1.000**	**1.000**	1.000		0.137	**1.000**	**1.000**	0.997		0.994	**1.000**	**1.000**	0.994		0.241	**1.000**	**1.000**	0.995	
C2H2	0.984	0.988	**0.996**	0.991		0.788	0.981	0.984	**0.999**		**0.966**	0.803	0.944	0.841		0.868	0.883	**0.963**	0.913	
C3H	0.987	0.994	**0.999**	0.996		0.709	0.994	0.984	**0.997**		0.962	0.797	**0.974**	0.875		0.817	0.885	**0.979**	0.932	
CAMTA	0.974	1.000	1.000	**1.000**		0.140	0.982	0.996	**0.996**		0.996	0.989	0.996	**1.000**		0.246	0.985	0.996	**0.998**	
CO-like	0.992	1.000	1.000	**1.000**		0.443	0.961	0.953	**0.991**		0.988	0.991	**0.995**	**0.995**		0.611	0.976	0.974	**0.993**	
CPP	0.991	**1.000**	1.000	**1.000**		0.349	0.979	0.944	**1.000**		0.988	**0.994**	**0.994**	0.972		0.516	**0.986**	0.968	0.986	
DBB	0.959	**0.997**	0.996	0.995		0.111	**0.953**	0.544	0.528		0.988	0.429	**0.997**	0.985		0.199	0.592	**0.704**	0.688	
Dof	0.998	1.000	**1.000**	1.000		0.901	0.990	**1.000**	**1.000**		0.989	0.995	**0.997**	0.996		0.943	0.993	**0.999**	0.998	
E2F/DP	0.993	1.000	1.000	**1.000**		0.442	0.994	0.975	**1.000**		0.975	0.983	**1.000**	0.994		0.608	0.989	0.988	**0.997**	
EIL	0.969	0.998	1.000	**1.000**		0.109	0.674	0.992	**1.000**		**0.984**	0.713	**0.984**	**0.984**		0.197	0.693	0.988	**0.992**	
ERF	0.993	**0.998**	0.995	0.997		0.910	**0.976**	0.929	0.958		0.985	0.996	0.996	**0.996**		0.946	**0.986**	0.961	0.977	
FAR1	0.984	0.997	0.998	**0.998**		0.602	0.881	0.933	**0.972**		0.979	**0.984**	0.975	0.936		0.745	0.930	0.953	**0.954**	
G2-like	0.985	**0.998**	0.993	0.992		0.673	**0.956**	0.819	0.806		0.973	0.974	**0.988**	0.987		0.795	**0.965**	0.896	0.887	
GATA	0.986	1.000	**1.000**	1.000		0.548	0.995	0.997	**0.998**		0.983	0.978	**0.996**	0.984		0.704	0.987	**0.997**	0.991	
GRAS	0.996	0.999	**0.999**	0.999		0.884	0.992	0.994	**0.998**		0.977	0.970	**0.987**	0.978		0.928	0.981	**0.990**	0.988	
GRF	0.998	1.000	**1.000**	1.000		0.744	0.997	**1.000**	**1.000**		0.997	0.995	**1.000**	0.997		0.852	0.996	**1.000**	0.999	
GeBP	0.962	**0.999**	0.999	0.998		0.111	**0.933**	0.829	0.694		0.958	0.939	**0.962**	**0.962**		0.198	**0.936**	0.891	0.806	
HB-PHD	0.993	1.000	1.000	**1.000**		0.182	0.969	**1.000**	**1.000**		**1.000**	0.990	0.990	**1.000**		0.308	0.979	0.995	**1.000**	
HB-other	0.967	**0.992**	0.984	0.979		0.165	**0.462**	0.294	0.224		**0.908**	0.664	0.886	0.798		0.279	**0.545**	0.442	0.350	
HD-ZIP	0.992	0.999	**0.999**	0.998		0.777	0.979	**0.982**	0.948		0.991	0.994	0.996	**0.998**		0.871	0.986	**0.989**	0.972	
HRT-like	0.943	1.000	1.000	**1.000**		0.013	0.980	**1.000**	**1.000**		0.960	**1.000**	0.980	**1.000**		0.026	0.990	0.990	**1.000**	
HSF	0.995	1.000	1.000	**1.000**		0.731	0.998	0.998	**1.000**		0.990	0.998	**0.999**	0.997		0.841	0.998	**0.998**	0.998	
LBD	0.978	**0.999**	0.999	0.999		0.511	**0.992**	0.975	0.981		0.979	0.982	**0.990**	0.979		0.672	**0.987**	0.982	0.980	
LFY	0.965	1.000	**1.000**	1.000		0.021	0.978	**1.000**	**1.000**		0.961	0.863	**0.980**	0.941		0.041	0.917	**0.990**	0.970	

Gene (Family)	Accuracy					Precision					Recall					F1				
	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend	k=2	k=3	k=4	k=5	trend
LSD	0.980	1.000	**1.000**	1.000		0.128	**0.984**	0.946	0.910		0.990	0.943	0.995	**1.000**		0.226	0.963	**0.970**	0.953	
M-type_MADS	0.945	**0.994**	0.984	0.994		0.292	**0.855**	0.599	0.807		0.953	0.906	**0.985**	0.972		0.447	0.880	0.745	**0.882**	
MIKC_MADS	[illegible]	**0.998**	0.995	0.994		0.811	**0.944**	0.829	0.777		0.996	0.986	0.996	**0.997**		0.894	**0.965**	0.905	0.874	
MYB	0.988	0.990	**0.997**	0.994		0.858	0.966	**0.973**	0.938		**0.993**	0.889	0.988	0.984		0.921	0.926	**0.981**	0.960	
MYB_related	[illegible]	**0.981**	0.977	0.974		0.314	**0.898**	0.707	0.676		**0.932**	0.691	0.896	0.866		0.470	0.781	**0.790**	0.759	
NAC	0.995	0.998	**0.999**	0.997		0.936	**0.998**	0.992	0.995		0.981	0.972	**0.988**	0.958		0.958	0.985	**0.990**	0.976	
NF-X1	0.988	1.000	**1.000**	1.000		0.092	0.953	**1.000**	**1.000**		**1.000**	**1.000**	**1.000**	0.975		0.169	0.976	**1.000**	0.988	
NF-YA	0.997	1.000	**1.000**	1.000		0.696	0.994	**0.996**	0.976		0.992	0.996	**0.998**	0.996		0.818	0.995	**0.997**	0.986	
NF-YB	0.978	**1.000**	1.000	1.000		0.306	**0.995**	0.989	0.970		0.974	0.992	**0.995**	0.994		0.465	**0.994**	0.992	0.982	
NF-YC	0.947	0.999	**1.000**	1.000		0.124	0.986	0.988	**0.998**		0.984	0.888	**0.992**	0.957		0.220	0.934	**0.990**	0.977	
NZZ/SPL	[illegible]	1.000	1.000	**1.000**		0.009	0.564	**1.000**	**1.000**		**1.000**	**1.000**	0.955	0.955		0.017	0.721	**0.977**	**0.977**	
Nin-like	0.997	0.999	0.999	**1.000**		0.717	0.960	0.936	**0.996**		**0.993**	0.946	0.971	0.955		0.833	0.953	0.953	**0.975**	
RAV	0.975	0.995	**0.996**	0.996		0.081	0.315	**0.374**	0.355		**1.000**	0.986	0.993	**1.000**		0.149	0.477	**0.544**	0.524	
S1Fa-like	0.932	**1.000**	1.000	1.000		0.016	0.986	**1.000**	**1.000**		**1.000**	0.986	0.958	0.958		0.032	**0.986**	0.979	0.979	
SAP	0.994	**1.000**	1.000	**1.000**		0.085	0.943	0.917	**0.970**		**1.000**	**1.000**	**1.000**	0.970		0.157	**0.971**	0.957	0.970	
SBP	0.969	0.999	1.000	**1.000**		0.293	0.999	0.975	**1.000**		0.976	0.942	**0.995**	0.984		0.450	0.970	0.985	**0.992**	
SRS	0.992	1.000	1.000	**1.000**		0.344	**1.000**	**1.000**	0.996		0.989	0.929	0.996	**1.000**		0.511	0.963	0.998	**0.998**	
STAT	0.980	**1.000**	**1.000**	**1.000**		0.032	**1.000**	**1.000**	**1.000**		**1.000**	**1.000**	**1.000**	**1.000**		0.063	**1.000**	**1.000**	**1.000**	
TALE	0.996	0.999	1.000	**1.000**		0.765	0.964	0.996	**0.999**		0.990	**1.000**	**1.000**	**1.000**		0.863	0.982	0.998	**0.999**	
TCP	0.978	1.000	**1.000**	1.000		0.368	0.994	0.995	**1.000**		0.982	0.993	**0.996**	0.990		0.536	0.993	**0.996**	0.995	
Trihelix	0.987	0.998	**0.999**	0.999		0.606	**0.993**	0.973	0.991		0.974	0.900	**0.979**	0.942		0.747	0.944	**0.976**	0.966	
VOZ	0.993	**1.000**	**1.000**	1.000		0.216	**1.000**	**1.000**	**1.000**		0.992	**1.000**	**1.000**	0.992		0.355	**1.000**	**1.000**	0.996	
WOX	0.964	1.000	1.000	**1.000**		0.169	0.985	0.981	**0.989**		0.985	**0.996**	**0.996**	**0.996**		0.288	0.991	0.988	**0.993**	
WRKY	0.995	0.999	1.000	**1.000**		0.906	0.992	0.996	**1.000**		0.986	0.987	**0.996**	0.995		0.944	0.989	0.996	**0.997**	
Whirly	0.944	1.000	1.000	**1.000**		0.028	0.981	0.991	**1.000**		0.981	**1.000**	**1.000**	0.991		0.054	0.991	**0.995**	0.995	
YABBY	0.967	1.000	1.000	**1.000**		0.140	0.920	0.968	**0.997**		0.994	**0.997**	0.980	0.983		0.246	0.957	0.974	**0.990**	
ZF-HD	0.992	1.000	**1.000**	1.000		0.502	0.963	**0.992**	0.981		0.967	**0.994**	**0.994**	0.992		0.661	0.978	**0.993**	0.987	
bHLH	0.987	0.987	**0.997**	0.996		0.884	**1.000**	0.991	0.992		**0.978**	0.850	0.967	0.959		0.929	0.919	**0.979**	0.975	
bZIP	0.996	0.994	**0.999**	0.999		0.948	0.996	0.994	**0.997**		0.975	0.885	**0.989**	0.979		0.961	0.937	**0.992**	0.988	

In [ ]:

In [ ]: